Universidad
Carlos III de Madrid

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

ORACLE ADF, UNA PRUEBA DE CONCEPTO

Autor: Diego Vélez Gil

Tutor: Miguel Ángel Patricio Guisado

Colmenarejo, 25 Junio de 2013

Agradecimientos

En primer lugar, quiero agradecer a mi tutor Miguel Ángel Patricio Guisado toda la ayuda que me ha ofrecido, no sólo en cuanto a este Trabajo de Fin de Grado, sino durante toda la carrera.

A mi familia, que me han ayudado a entender la importancia de las cosas y a ser una mejor persona. Siempre han estado ahí para cualquier problema que tuviera e incluso priorizándolos sobre los suyos propios.

A mi novia, que durante este último año me ha dado su inestimable apoyo y ha sido capaz de aguantar los malos tragos que he pasado.

Y por último a todos mis amigos, a los de toda la vida, a los de la universidad y los que conocí en Alemania, que han estado ahí para los buenos y los malos momentos, y han hecho que estos últimos años de universidad sean unos de los más importantes de mi vida.

Gracias a todas estas personas que me han ayudado a que concluya una etapa muy importante en mi vida y que seguro me ayudaran a las venideras de los próximos años.

Resumen

Este Trabajo Fin de Grado pretende realizar un análisis sobre el framework de desarrollo de aplicaciones J2EE, Oracle ADF Framework. Para ello se ha se ha realizado un estudio de cómo se desarrolla cada una de las capas en las que está separada una aplicación J2EE (Modelo-Vista-Controlador) mediante este framework y que alternativas de implementación proporciona para cada una de ellas.

Por otro lado se han mostrado las características de los principales frameworks de desarrollo de aplicaciones J2EE, una comparativa de las principales características y los beneficios que ofrece Oracle ADF frente a ellos.

Además se ha explicado la siguiente generación de aplicaciones empresariales de Oracle, que tecnologías la forman y el rol de ADF respecto a éstas. JDeveloper es el entorno de desarrollo diseñado por Oracle para implementar aplicaciones con ADF, por lo que se ha mostrado mediante ilustraciones la forma de utilizar el JDeveloper para implementar una aplicación ADF y a utilizar las principales herramientas que proporciona.

Para afianzar conocimientos obtenidos en las etapas previas de este Trabajo de Fin de Grado se ha desarrollado una prueba de concepto con ADF Framework, en la que se ha mostrado la forma de diseñar una aplicación con respecto a las capacidades que ADF ofrece y su implementación con JDeveloper, a partir de unos requisitos previos.

Por último se ha desarrollado un manual para facilitar al usuario el uso de la prueba de concepto desarrollada.

Índice de Contenidos

Índice de ilustraciones

Oracle ADF, una prueba de concepto

1. Introducción

1.1. Ambientación

Con el desarrollo de la tecnología las necesidades de las empresas han aumentado sus necesidades y a su vez las aplicaciones no sólo requieren más funcionalidad, sino que además deben ser más escalables y deben ofrecer sus servicios de manera global.

J2EE es una solución tecnológica a esta premisa, y que además está ganando cada vez más importancia. Muestra de ello es la gran cantidad de frameworks existentes y sobretodo la apuesta de grandes empresas como Oracle.

1.1.1. J2EE

Java EE (Java Enterprise Platform) es una plataforma de programación para desarrollar aplicaciones en el lenguaje java, que está diseñada cumpliendo los estándares y para ayudar a los desarrolladores a crear aplicaciones web escalables, seguras y multicapa.

En la actualidad Java EE forma parte de la base de muchas aplicaciones empresariales y provee un conjunto de especificaciones que están construidas sobre Java SE (Java Standard Edition). Generalmente los requisitos de estas aplicaciones (seguridad, fiabilidad, usabilidad) aumentan increíblemente la complejidad, y por ello Java EE provee un modelo de desarrollo, una API (Application Programming Interface) y un entorno de ejecución que permite a los desarrolladores que se concentren en la funcionalidad.

Como norma en las aplicaciones, la funcionalidad de la aplicación está separada en distintas áreas llamadas capas. El ejemplo típico es el modelo MVC (modelo-vista-controlador).

El modelo MVC separa los datos y la lógica de negocio de la interfaz del usuario en tres capas, su objetivo más importante es la reutilización de código, pero además se consigue una separación de conceptos que facilitan las tareas de desarrollo y de mantenimiento del código.

Las capas MVC se pueden definir como:

- **Modelo:** es la representación específica de la información con la que el sistema opera. Es responsable de acceder a la capa de almacenamiento de datos y lo ideal es que sea independiente del sistema de almacenamiento.
- **Vista:** presenta el modelo en un formato adecuado para interactuar, usualmente con el usuario.
- **Controlador:** es el responsable de recibir los eventos de entrada y gestionar las respuestas a dichos eventos en función de unas reglas preestablecidas.

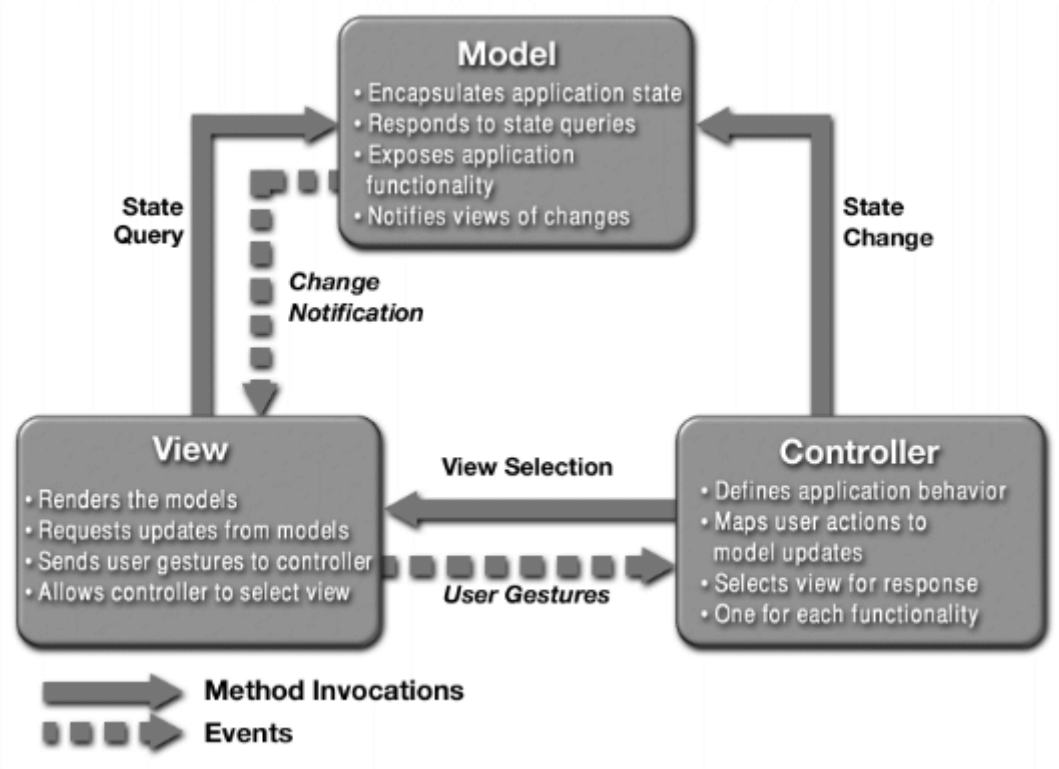

Ilustración 1. Esquema modelo MVC

El diseño modular permite además a los diseñadores y a los desarrolladores trabajar conjuntamente, pudiéndose realizar cambios en las distintas capas sin afectar a las otras.

Aunque se puedan encontrar diferentes implementaciones de MVC, el flujo de control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario (por ejemplo, el usuario pulsa un botón)
2. El controlador recibe la notificación del evento generado (por los objetos de la vista) por la acción del usuario, y lo gestiona generalmente a través de un manejador de eventos.

3. El controlador accede al modelo para recuperar o modificar los datos de la aplicación.
4. Una vez el controlador ha realizado las operaciones necesarias en el modelo de datos, se delega a la vista la tarea de presentar los mismos al usuario.
5. La interfaz de usuario pasa a un estado de espera de interacciones del usuario.

Con el desarrollo y expansión de internet, las empresas han habilitado sus servicios a la web, y como resultado los servidores de aplicación han dejado de encargarse de la tarea de la capa de presentación y se ha trasladado a servidores web especializados. Por lo que el modelo MVC evolucionó en el modelo multicapa, en el que cada capa está separada por la red pero sigue manteniéndose la separación de contenidos del modelo MVC.

Los servidores de aplicación alojan componentes de negocio o servicios que son accesibles remotamente, así distintas aplicaciones pueden acceder al mismo componente de manera paralela e independiente. Con esta separación de la aplicación en componentes remotamente accesibles, los desarrolladores solamente tienen que modificar o añadir un componente o capa específica, en vez de crear una nueva completa aplicación, si deciden cambiar tecnologías o aumentar su escalabilidad.

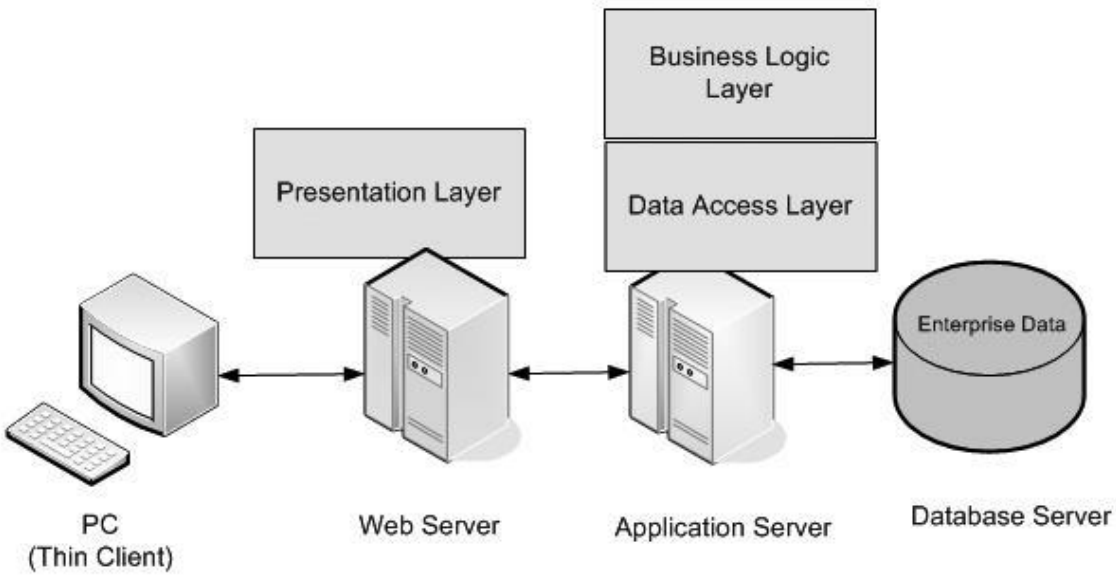

Ilustración 2. Esquema modelo multicapa

Distribuir el procesamiento en capas separadas mejora la utilización de recursos y además ayuda a la distribución de las tareas a expertos que están mejor formados para trabajar y desarrollar en una determinada capa. Por ejemplo los

desarrolladores web están más experimentados para trabajar con la capa de presentación en el servidor web mientras que los desarrolladores de bases de datos lo están para trabajar la capa de datos. Mantener estas capas aisladas de las demás no tiene ningún propósito útil, sino que deben estar completamente integradas para conseguir el objetivo empresarial.

Por otra parte, las aplicaciones distribuidas necesitan varios servicios, y por ello deben ser capaces de gestionar o realizar transacciones mientras interactúan con distintos sistemas de información. Debido a que estas aplicaciones multicapa son accedidas a través de internet está característica es muy importante para asegurar la concurrencia y por el mismo motivo deben tener implementados robustos mecanismos de seguridad para asegurar los datos.

En Java EE se utiliza este modelo multicapa anteriormente explicado, y las capas están separadas en capa cliente (client tier), capa intermedia (middle tier) y capa de datos (data tier). El desarrollo de aplicaciones Java EE se centrar en la capa intermedia.

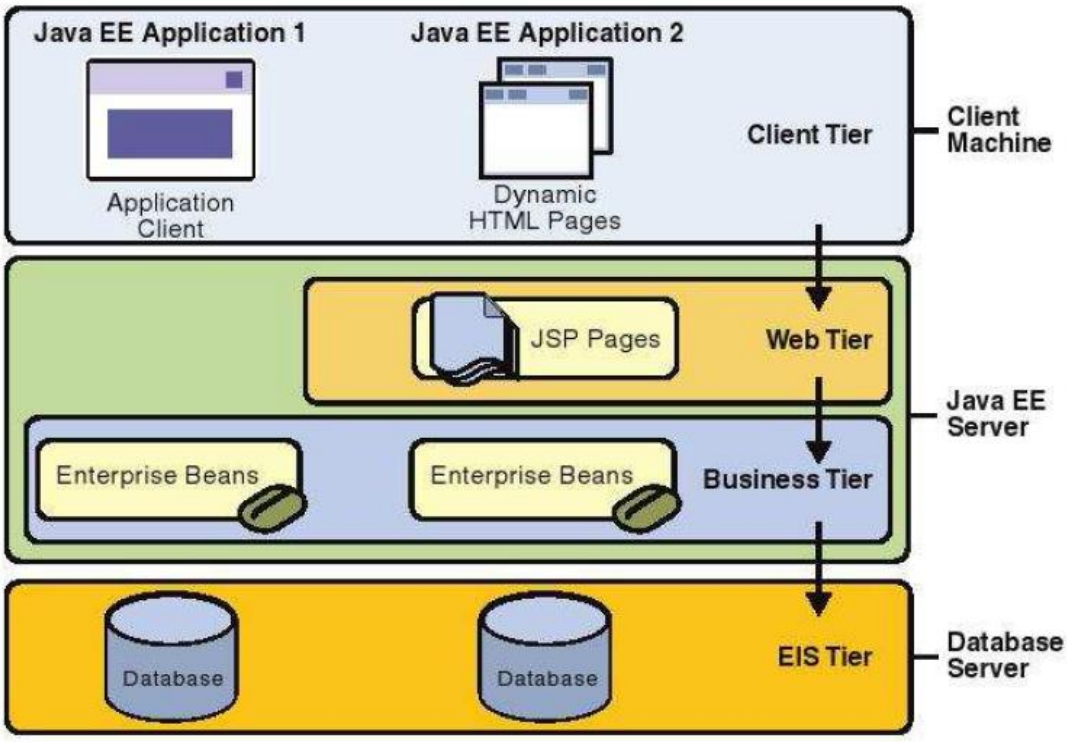

Ilustración 3. Esquema general Java EE

- **Capa cliente:** consiste en clientes de aplicación que acceden al Java EE Server y por norma general están en una maquina diferente que el servidor. El cliente realiza peticiones al servidor que éste procesa y devuelve una respuesta al cliente. Hay muchos tipos diferentes de aplicaciones que pueden ser clientes Java EE y no siempre tienen que ser aplicaciones Java, sino que pueden ser navegadores web, aplicaciones independientes u otros servidores.

- **Capa intermedia:** la capa intermedia está alojada en el servidor Java EE y está a su vez separada en dos:

    - **Capa web:** está compuesta por componentes que manejan la interacción entre los clientes y la capa de negocio. Entre sus funciones esta generar contenido en distintos formatos para el cliente, mantener los datos de la sesión del usuario, controlar el flujo entre páginas en el cliente o recoger los eventos de la interfaz del cliente.

Web-Tier Java EE Technologies

Technology	***Purpose***
Servlets	*Java programming language classes that dynamically process requests and construct responses, usually for HTML pages*
JavaServer Faces technology	*A user-interface component framework for web applications that allows you to include UI components (such as fields and buttons) on a page, convert and validate UI component data, save UI component data to server-side data stores, and maintain component state.*
JavaServer Faces Facelets technology	*Facelets applications are a type of JavaServer Faces applications that use XHTML pages rather than JSP pages.*
Expression Language	*A set of standard tags used in JSP and Facelets pages to refer to Java EE components.*
JavaServer Pages (JSP)	*Text-based documents that are compiled into servlets and define how dynamic content can be added to static pages, such as HTML pages.*
JavaServer Pages Standard Tag Library	*A tag library that encapsulates core functionality common to JSP pages*
JavaBeans Components	*Objects that act as temporary data stores for the pages of an application*

    - **Capa de negocio:** está compuesta por los componentes que proveen la lógica de negocio a la aplicación. La lógica de negocio esta forma por código que provee una funcionalidad particular a un determinado dominio.

- **Capa datos:** esta capa está compuesta por servidores de bases de datos y el acceso estas y generalmente se llama capa EIS (Enterprise Information Systems). Este tipo de recursos al igual que las aplicaciones clientes están alojados en distintas máquinas que el servidor.

EIS Tier Java EE Technologies

Technology	***Description***
The Java Database Connectivity API (JDBC)	*A low-level API for accessing and retrieving data from underlying data stores. A common use of JDBC is to make SQL queries on a particular database.*
The Java Persistence API	*An API for accessing data in underlying data stores and mapping that data to Java programming language objects. The Java Persistence API is a much higher-level API than JDBC, and hides the complexity of JDBC from the user.*
The Java EE Connector Architecture	*An API for connecting to other enterprise resources, like enterprise resource planning or customer management system software.*
The Java Transaction API (JTA)	*An API for defining and managing transactions, including distributed transactions or transactions that cross multiple underlying data sources.*

1.1.2. Frameworks J2EE

Un framework en software es una estructura que sirve de soporte o de guía para el desarrollo de proyectos que siguen los estándares de una determinada tecnología. Además proveen un conjunto de programas, bibliotecas y lenguaje interpretado para unir los diferentes componentes de un proyecto.

En la actualidad hay una gran cantidad de frameworks J2EE y cada uno de ellos tiene características propias, simplificando un aspecto del desarrollo distinto. Esto supone que el desarrollo pueda ser totalmente distinto en frameworks distintos.

Los frameworks más importantes en la actualidad son:

- JSF (Java Server Faces)
- Struts
- Spring
- J Boss Seam
- GWT (Google Web Toolkit)

A continuación se van a describir sus características más importantes.

1.1.2.1. Java Server Faces

JSF es un framework de interfaces de usuario del lado de servidor para aplicaciones Web basadas en J2EE desarrollado por la Java Community Process como

JSR 127. Para desplegar páginas usa Java Server Pages (JSP) y puede ser integrado con las bibliotecas de componentes de JSF JBoss RichFaces o ICEFaces.

Java Server Faces proporciona una API para componentes de la interfaz del usuario, en los que se administra su estado, maneja eventos, valida entradas, define navegación entre distintas páginas y da soporte para internacionalización y accesibilidad. Además de la API existe una librería de etiquetas JSP personalizadas para mostrar componentes UI dentro de una página JSP.

Este modelo de programación bien definido ayuda de forma significativa la implementación y mantenimiento en la parte del servidor. La interfaz de usuario se ejecuta en el servidor y posteriormente se renderiza en el cliente.

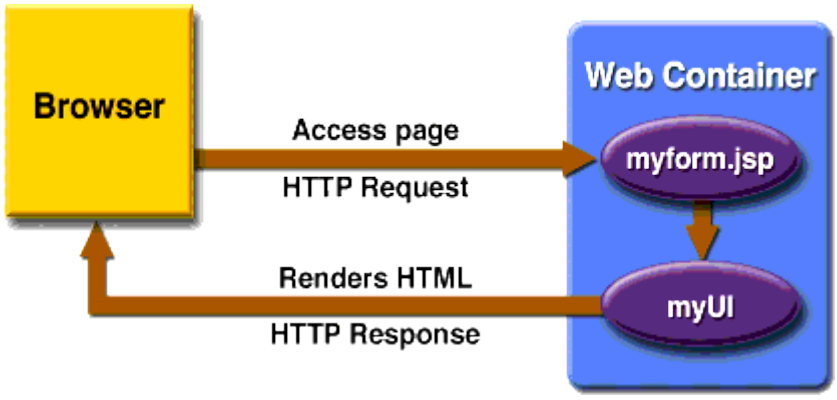

Ilustración 4. Ciclo de ejecución JSF

En la figura superior el navegador web envía una petición HTTP al servidor web, la página JSP recibe la petición y dibuja los componentes de la interfaz de usuario (también crea los manejadores de eventos del cliente) con las etiquetas personalizadas definidas por JSF). Se envía un respuesta HTTP al cliente con el código HTML renderizado.

La principal ventaja que ofrece la tecnología Java Server Faces es que ofrece una clara separación entre la funcionalidad y la presentación. Las aplicaciones JSP no pueden mapear peticiones HTTP al manejo de eventos específicos de componentes o manejar elementos UI como objetos con estado en el servidor, por lo que con JSF se permite la construcción de aplicaciones web con esta separación.

1.1.2.2. Struts

Es un framework "open-source" de desarrollo de aplicaciones J2EE basado en el modelo MVC, en el que la lógica de la aplicación se ejecuta en Java Servlets. Struts fue desarrollado en un principio como parte del proyecto Apache Jakarta, y posteriormente se consolidó como un proyecto en sí.

Es importante separar las dos versiones que existen de Struts, ya que aunque generalmente las segundas versiones de una tecnología son la evolución de la primera, este no es el caso y Struts 2 es una evolución del framework Webwork pero manteniendo las ventajas de Struts 1.

Apache Foundation ha anunciado que Struts 1 ha llegado a su fin y no ofrece más soporte oficial, por lo que se explicarán únicamente las características de Struts 2.

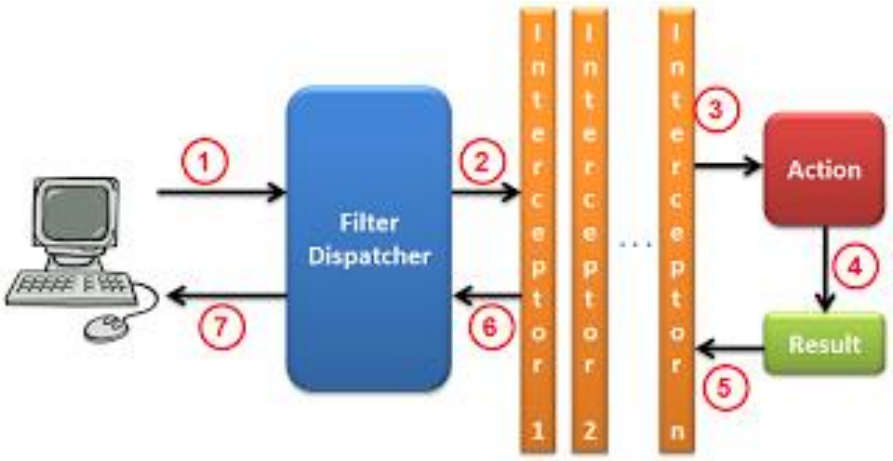

Ilustración 5. Esquema ciclo de ejecución Struts 2

Como se puede ver en la ilustración los 3 componentes principales son el "*Dispatcher Filter*", los "*Interceptors*", los "*Actions*" y los "*Results*".

- **Dispatcher Filter** es un filtro que se encarga de comenzar la ejecución de la cadena de interceptores y ejecutar los "*actions".*
- **Los Interceptores** son clases que permiten que se implementen funcionalidades cruzadas o comunes para todos los "*actions*". Realizan tareas antes y después de que se ejecuten los "*actions",* por lo que en general sirven para ejecutar algún proceso particular relativo a estos.
- **Las Acciones** son clases encargadas de realizar la lógica referente a una determinada petición. Cada URL es mapeada con una acción específica y usualmente devuelven los denomidados "*results*".
- **Los Resultados** son las respuestas que devuelven los "*actions"* y están

compuestos por el "*result*" y el tipo del result (indica como debe ser tratado por los interceptores)

La principal ventaja de Struts es que reduce la complejidad de desarrollo del controlador y permite centrarse en la lógica de negocio (en los "*actions*").

1.1.2.3. Spring

En la actualidad Spring es el framework de desarrollo de aplicaciones J2EE más utilizado y fue creado por Rod Johnson como solución y alternativa a la laboriosa tarea de trabajar con las primeras versiones de EJB.

La base de Spring es la inyección de dependencias. Al diseñar una aplicación en J2EE dispones de una gran cantidad de objetos que se relacionan entre sí, por ello en Spring se declaran las dependencias de los componentes en un XML y él se encarga de instanciar los objetos y las relaciones entre ellos. Con esto se consigue evitar el acoplamiento que puede surgir en grandes aplicaciones, en las que las relaciones y dependencias entre los distintos componentes que no es un tema trivial.

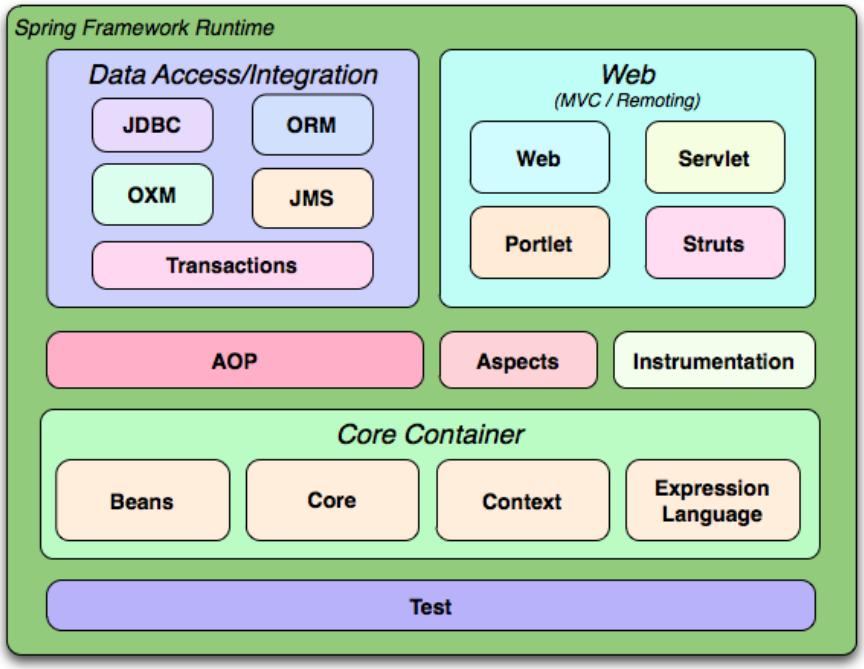

Ilustración 6. Módulos del framework Spring

Spring tiene un tamaño considerado y por ello está divido en distintos módulos.

En la ilustración se pueden observar los 20 módulos por los que Spring está formado.

- **Core Container:** está compuesto por el módulo de Core, Beans, Context y Expression Language y sus características son:
  - Core: provee la parte fundamental del framework, se encarga de la inyección de dependencias.
  - Beans: provee la "*BeanFactory*" que es una sofisticada implementación de "*Factory Method*".
  - Context: es un medio de acceso a los objetos definidos y configurados, el foco del módulo es la interfaz "*Application Context*".
  - Expression Language: modulo que provee un potente Expression Language para consultar y manipular objetos en ejecución.
- **Data Access/Integration:** está compuesto por el módulo de JDBC, ORM, OXM, JMS y Transactions y sus características son:
  - JDBC: provee una capa de abstracción JDBC que evita el tedioso código relacionado con JDBC.
  - ORM: provee integración para las APIs de mapeo objeto-relacional (JPA, JDO, Hibernate e iBatis).
  - OXM: provee integración para implementaciones de mapeo objeto-XML (JAXB, Castor, XMLBeans, JiBX y XStream).
  - JMS: Java Messaging Service, contiene funcionalidades para producir y consumir mensajes de las colas.
  - Transactions: gestiona transacciones programáticas y declarativas para clases que implementan interfaces especiales y para los POJOs.
- **Web:** está compuesto por el módulo de Web, Servlet, Portlet y Struts y sus características son:
  - Web: provee las características básicas orientadas a la web, tales como subir un archivo "*multipart*" o inicializar el contenedor loC usando los "*servle-listeners*" y el "*application-context*".
  - Servlet: contiene la implementación MVC para las aplicaciones web.
  - Porlet: provee implementación MVC para un entorno con portlets.
  - Struts: provee integración para la capa web de Struts dentro de una aplicación Spring.
- **Otros:** son el resto de módulos(AOP, Aspects, Instrumentation y Test)y sus características son:
  - AOP: provee la implementación de la programación orientada a aspectos, permitiendo definir interceptores y "*pointcuts"* para

separar funcionalidades.

- Aspects: modulo que provee integración con AspectJ que es un maduro y potente framework de programación orientada a aspectos
- Intrumentation: provee utlidades que pueden ser aplicadas en ciertos servidores de aplicaciones.
- Test: provee testeo de componentes de Spring con frameworks como TestNG o JUnit.

Las grandes ventajas de Spring son que ofrece un framework para cada una de las capas de la aplicación, además no existen "*ActionForms*" ya que se enlaza directamente con los beans de negocio y produce una gran cantidad de código testeable.

1.1.2.4. JBoss Seam

Seam es un potente framework de desarrollo de aplicaciones empresariales siguiendo el estándar J2EE. Integra distintas tecnologías como JSF y EJB 3.0, y además AJAX, JPA y BPM (Bussiness Process Management) en una única plataforma.

Seam ha sido diseñado para eliminar la complejidad a nivel de arquitectura y de APIs, y permite a los desarrolladores a ensamblar complejas aplicaciones web usando simples clases javas con anotaciones, un set de componentes UI y una pequeña parte XML, todo ello a través de unas sofisticadas herramientas llamadas JBoss Tools.

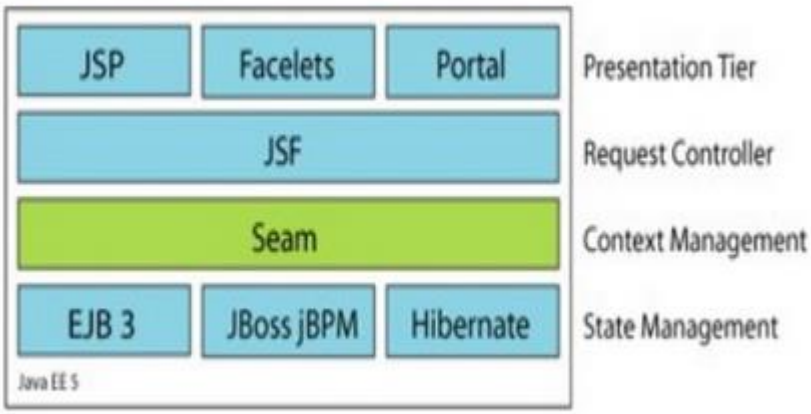

Ilustración 7. Arquitectura JBoss Seam

Seam elimina la barrera existente entre JSF y EJB, permitiendo usar EJBs directamente como "*Backing Bean*" de JSF y lanzar o escuchar eventos.

1.1.2.5. Google Web Toolkit

GWT es un framework open source de desarrollo de aplicaciones J2EE en el que Java es el único lenguaje utilizado y el compilador de GWT lo convierte a JavaScript y HTML.

GWT está compuesto principalmente por cuatro componentes:

- **Compilador GWT Java-JavaScript:** se encarga de generar el código JavaScript a partir del código Java desarrollado y se utiliza cuando se ejecute la aplicación en modo web.
- **Navegador web "*hosted*":** permite ejecutar aplicaciones GWT en modo "*hosted*", es decir, se ejecuta el código Java sobre una máquina virtual sin ser traducido a JavaScript.
- **Emulación de librerías JRE:** contiene las implementaciones en JavaScript de las librerías más usadas en Java(principalmente los paquetes "*java.lang*" y "*java.util*")
- **Librería de clases de interfaz de usuario:** son un conjunto de interfaces y clases personalizadas que permiten desarrollar "*widgets*" para los navegadores.

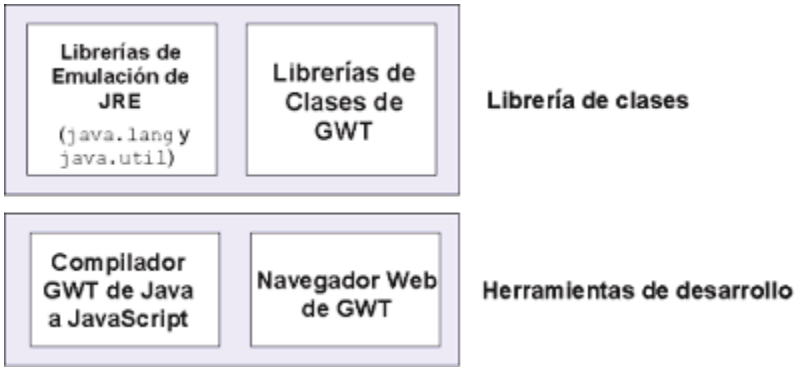

Ilustración 8. Arquitectura de Google Web Toolkit

El framework GWT tiene una metodología de desarrollo totalmente distinta a los demás frameworks J2EE que permite a los desarrolladores especializarse en un único lenguaje. Además el código JavaScript generado por el compilador GWT está

altamente optimizado y puede ser ejecutado en todos los navegadores, incluidos los integrados en dispositivos móviles.

Con GWT gran parte de la lógica de la interfaz puede ser transferida al lado del cliente, obteniéndose un mejor rendimiento y un menor consumo de ancho de banda al evitar el constante envío de datos al servidor. El mecanismo de comunicación en GWT con el servidor es mediante RPC (Remote Procedure Call) y permite enviar objetos entre el servidor y el cliente (puede manejar el polimorfismo e incluso excepciones).

1.1.2.6. Tabla Comparativa de los distintos frameworks

Project	Language	Ajax	MVC	ORM	Testing	Template	Cache	Form Validation
JSF	Java	Si	Si	Si, con extensiones	JUnit	Facelets, JSP	Si	Native Validators, integración con Bean Validators
Struts	Java	Si	Si	Si	Unit Tests	Si		Si
Spring	Java	Si	Si	Hibernate, iBatis y otros	Objetos Mock y Unit Test	JSP, Common Tiles, Velocity, Thymeleaf y otros	Ehcache y otros	Commons Validator, Bean Validator
Seam	Java	Si	Si	JPA e Hibernate	JUnit y TestNG	Facelets	JBoss Cache y Ehcache	Hibernate Validator
GWT	Java, JavaScript	Si		Request Factory	JUnit, jsUnit y Selenium			Bean Validation

1.1.3. Oracle Application Depevelopment Framework

Oracle Application Development Framework, más conocido como Oracle ADF, es un framework de desarrollo de aplicaciones empresariales J2EE propietario de Oracle (tiene una versión limitada gratuita llamada "*ADF Essentials*").

Oracle ADF simplifica el desarrollo de Java EE minimizando la necesidad de escribir código que implementa la arquitectura de la aplicación permitiendo a los desarrolladores centrarse los requisitos de ésta. Oracle ADF implementa el patrón de diseño MVC y ofrece una solución integrada que cubre todas las capas de esta arquitectura (ORM, persistencia de datos, reutilización de código de la capa del controlador, interfaz de usuario, enlace de datos con la interfaz de usuario y la seguridad). Además el incremento de la necesidad de las empresas de construir aplicaciones que utilicen los principios de "*Service Oriented Architecture (SOA)*", ha llevado a Oracle a integrar en ADF estas características a través de Oracle SOA y "*Webcenter Portal*".

Las características clave de Oracle ADF que lo sitúan por encima de otros frameworks J2EE son:

- **Soluciones End-to-End.** Oracle ADF no se centra en una única capa de la arquitectura de Java EE, sino que provee soluciones todas ellas, desde la capa de la vista hasta la de acceso a datos.
- **Entorno de desarrollo.** Otros frameworks carecen de robustas herramientas desarrollo, en cambio Oracle JDeveloper es la herramienta de desarrollo de Oracle ADF y provee ayudas visuales y un enfoque declarativo que minimiza la necesidad de escribir código. Además para los desarrolladores que prefieran usar otro IDE, existe un paquete para Eclipse que integra todas las características ("*Oracle Enterprise Pack for Eclipse*").
- **Independencia de la plataforma.** Otros frameworks están centrados en un solo proveedor de software, sin embargo, "*ADF Runtime*" puede ser instalado en varios servidores de aplicaciones que soportan J2EE y servicios de negocio que pueden conectar con cualquier base de datos basada en SQL-92.
- **Elección de las tecnologías.** Los desarrolladores generalmente tienen preferencias a la hora de implementar una capa de la aplicación, por ello Oracle ADF soporta múltiples tecnologías para cada una de las capas de las aplicaciones y no fuerza a usar una tecnología o un estilo de desarrollo determinado.
- **Compromiso con la tecnología.** Es importante recalcar que Oracle ADF es la tecnología elegida por Oracle para la siguiente generación de aplicaciones empresariales (Oracle Fusion) y está en continuo uso interno por parte de Oracle (siguiendo el dicho de "Eat your own dog food").
- **Basado en metadatos.** Oracle ADF permite la opción de desarrollar cada una de las capas del framework de manera declarativa, a través de

ficheros XML.

- **Customización declarativa.** Oracle ADF permite a las organizaciones usar una aplicación base y customizarla para cumplir los requisitos de los distintos usuarios. En conjunción con MDS (Servicios de Metadatos) la customización puede ser realizada a todos los usuarios de la aplicación que pertenecen a un grupo determinado o la llamada "personalización" en la que el usuario realiza las customizaciones para mejorar su experiencia personal.
- **Reusabilidad mejorada.** JDeveloper junto con ADF proveen características que fomentan la reusabilidad de código (plantillas JSF, plantillas task flows, servicios de negocio, librerías ADF, regiones basadas en JSF fragments, y muchas otras).
- **Soporte.** Oracle ADF es un producto oficial de Oracle y por ello ofrece soporte inmediato a las organizaciones.

1.2. Motivación

En la actualidad se ha incrementado de manera notable la necesidad de las empresas de crear aplicaciones distribuidas, transaccionales y portables que aprovechen la velocidad, seguridad y la fiabilidad de la tecnología que los servidores proporcionan. En el mundo de tecnología de la información, estas aplicaciones tienen que ser diseñadas, construidas y producidas por menos dinero, mayor velocidad y con menos recursos. El objetivo de los frameworks es precisamente ayudar a los desarrolladores a conseguir estas premisas.

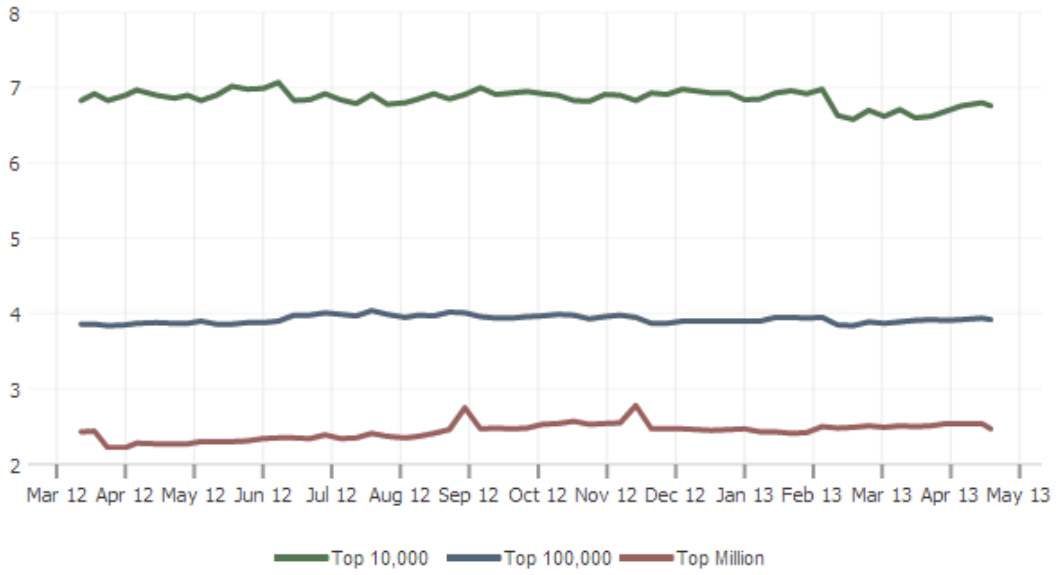

Ilustración 9. Porcentaje de uso de J2EE

Como se ha mostrado antes en la actualidad hay una gran cantidad de frameworks J2EE distintos, cada uno con sus beneficios y características propias. Entre todos ellos Oracle ADF ha sido el framework elegido por todo el conjunto de razones que se exponen a continuación.

De acuerdo con Oracle, más de 30.000 organizaciones han implantado Oracle Fusion Middleware, incluyendo 35 de las 50 mayores empresas de la actualidad y más de 750 de las 1000 de "*Business Week Global*", y además cuenta con el soporte de más de 7500 partners. Estas cifras nos indican que en la actualidad Oracle Fusion Middleware cuenta con una gran cuota de mercado y la herramienta con la que se desarrollan aplicaciones de Oracle Fusion Middleware es ADF framework.

A la hora de decidirse por un determinado framework, un factor de gran importancia es si la inversión que se va realizar está asegurada al menos a medio-largo plazo. La apuesta de Oracle para el futuro es ADF y muestra de ello es que todas las aplicaciones internas de la compañía han sido migradas a esta tecnología. Ninguna tecnología está libre de errores (y ADF no es una excepción) pero este hecho supone que todo el equipo de desarrolladores de Oracle actúe como probador y ayude al descubrimiento y solución de errores, dando una estabilidad y calidad bastante notables respecto a otros frameworks.

Toda aplicación web contiene una parte de la funcionalidad realmente avanzada y compleja que debe ser realizada en Java puro. Para esta parte que suele estar entre un 5-20% la productividad es común a la de los demás frameworks, sin embargo para el resto de la funcionalidad ADF proporciona una gran numero de mecanismos "out of the box" y un gran porcentaje de las funcionalidades más comunes deseadas se realizan de manera declarativa, con lo que incluso programadores sin experiencia en java pueden crear grandes contribuciones en el desarrollo gracias al enfoque de programación visual que proporciona el JDeveloper.

Además recientemente Oracle ha lanzado una extensión para el JDeveloper llamada ADF Mobile, que permite a los desarrolladores construir aplicaciones empresariales para dispositivos móviles a partir de un único código base. Y lo más importante es que el desarrollo de éstas es muy similar al de una aplicación común de ADF.

A todo lo descrito anteriormente, he de añadir mi interés personal en aprender el desarrollo de aplicaciones en ADF framework, que supondrá una nueva habilidad que mejora mi currículo de cara al mundo laboral.

1.3. Objetivos

El objetivo principal de este trabajo es el análisis del framework de desarrollo de aplicaciones Java EE Oracle ADF. Con este objetivo se pretende conocer la capacidad de desarrollo de aplicaciones en JDeveloper, la productividad a la hora de desarrollar una aplicación y el resultado final de una aplicación haciendo uso de ADF. Así mismo, se plantean otra serie de objetivos secundarios:

- Comparar el desarrollo de aplicaciones en el framework ADF respecto a su antigua versión (OAF framework).
- Descubrir el esfuerzo que se necesita para aprender a desarrollar aplicaciones en este framework
- Desarrollar una prueba de concepto, con el fin de afianzar los conocimientos resultantes del análisis de ADF y la aplicación práctica de los mismos.
- Realizar una toma de contacto con Weblogic, que es el servidor de aplicaciones por defecto de ADF y al que se están migrando las aplicaciones Oracle.

1.4. Medios

Los recursos utilizados para el análisis del framework ADF son muy reducidos, simplemente con un ordenador capaz de ejecutar el JDeveloper, que según las especificaciones de Oracle debe ser Pentium IV 2GHz o superior y 2 GB de RAM. Aunque con esas características sería prácticamente imposible realizar ningún proyecto, vamos a tomarlas como válidas ya que en esta parte sólo se realizará tareas de investigación.

Los recursos necesarios para desarrollar la prueba de concepto son mayores, ya que si se precisan tareas de desarrollo, y además un conjunto de requisitos (servidor independiente y autenticación a través de un LDAP).

El desarrollo de la prueba de concepto necesita en un principio un ordenador de gran potencia (el utilizado ha sido un HP Pavilion Intel Core i3 con 4 GB de RAM) para poder ejecutar el servidor de Weblogic que JDeveloper tiene integrado.

Posteriormente, para que la aplicación esté disponible se necesita un servidor Weblogic en una máquina independiente y otra máquina con un LDAP instalado con el que interactuará el servidor Weblogic para que se encargue de la autenticación de los usuarios.

1.5. Contenido del documento

Dentro de este punto, se listan los distintos capítulos que componen el documento, así como una descripción del contenido:

- **Capítulo 1, Introducción:** determina la motivación por la que se ha realizado este Trabajo de Fin de Grado, los objetivos que se quieren conseguir y una pequeña ambientación.
- **Capítulo 2, Tecnologías para el desarrollo de la nueva generación de aplicaciones empresariales en Oracle:** explica la nueva generación de aplicaciones empresariales, como encaje ADF en ellas, la arquitectura de ADF, JDeveloper y Weblogic
- **Capítulo 3, Conceptos básicos en el de una nueva aplicación con Oracle ADF framework y Oracle JDeveloper:** describe como desarrollar aplicaciones con ADF Framework de una manera práctica
- **Capítulo 4, Prueba de Concepto:** explica cómo se han aplicado los conceptos de ADF durante la realización de un proyecto
- **Capítulo 5, Conclusiones:** contiene un resumen que muestra el cumplimiento de los requisitos y las posibles líneas de desarrollo posteriores a este trabajo
- **Capítulo 6, Planificación y presupuesto:** Detalla el calendario de dedicación al Trabajo Fin de Grado y las modificaciones sufridas. También el cálculo de del coste global de la realización del Trabajo de Fin de Grado.
- **Capítulo 7, Requisitos de Usuario:** Las especificaciones que debe cumplir la prueba de concepto que se ha realizado.
- **Capítulo 8, Manual de Usuario:** una pequeña guía de uso de la prueba de concepto.

2. Tecnologías para el desarrollo de la nueva generación de aplicaciones empresariales en Oracle

2.1. Oracle Fusion

Generalmente la mayoría de la gente asocia el nombre de Oracle con el término "bases de datos" (y siendo claros, Oracle ha desarrollado gran parte de su negocio en este campo y sigue siendo su principal elemento), pero según la tecnología ha avanzado, y por lo tanto la necesidad de las empresas, Oracle ha tenido que adaptarse y en la actualidad ofrece distintas alternativas de desarrollo de aplicaciones al lenguaje PL/SQL.

La evidencia clara es el nuevo término que nació en el 2006: *Fusion*.

Fusion, en realidad no es un objeto tangible, ni es un producto o una tecnología específica, sino que era el nombre originalmente de "*Project Fusion"*, y ha evolucionado a un término de marketing para designar al conjunto de aplicaciones de negocio de Oracle y sus correspondientes tecnologías.

Como ya se ha dicho parte del éxito de Oracle es atribuible a las bases de datos, y según crecía el negocio, Oracle lo fue complementando con herramientas de desarrollo como "*Oracle Forms"* y "*Oracle Reports",* y con un propio conjunto de aplicaciones de negocio (Oracle E-Business Suite). Este crecimiento convirtió a Oracle en una gran empresa, y a partir de 2004 puso en marcha una serie de grandes adquisiciones que comenzó con "*PeopleSoft*", y seguidamente "*Siebel Systems"* en 2005. La adquisición de ambas compañías no solo significo nuevos empleados y clientes, sino también significantes productos que añadir a los ya existentes. Luego siguieron las adquisiciones con "*Hyperion Solutions", "BEA Systems"* y *"Sun",* que llevaron a Oracle tecnologías líderes en el mercado.

Oracle se encontró con un significante conjunto de tecnologías, herramientas y aplicaciones de negocio. Así que Oracle fijó como próximo objetivo crear una nueva generación de aplicaciones que aprovecharan todo el conjunto, pero para ello no se podía simplemente juntar las mejores características de cada tecnología y esperar que el resultado fuera la siguiente generación, ya que cada una estaba implementada usando distintas tecnologías, lenguajes de programación y herramientas de desarrollo. Por lo que Oracle comenzó el diseño de una nueva plataforma en la que se integrarán todas las tecnologías, éstas tuvieron que ser estandarizadas y modernizadas para hacerlas compatibles entre sí, surgiendo "*Project Fusion".*

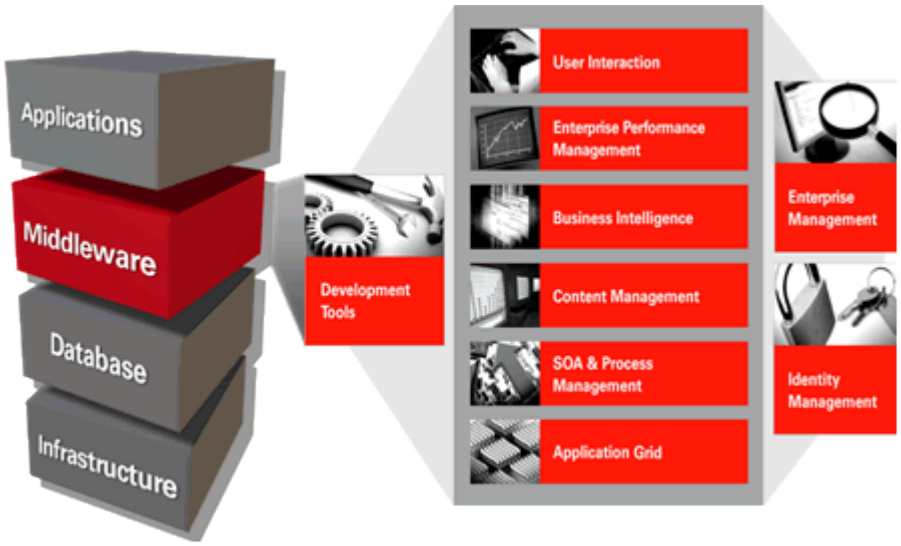

Ilustración 10. Esquema Oracle Fusion

Dentro del término Fusion hay que destacar tres pilares fundamentales:

- **Oracle Fusion Applications.** La siguiente generación de aplicaciones de negocio de Oracle están agrupadas bajo este nombre. Estas son distribuidas como módulos y los clientes las adquieren en función de sus necesidades. Más allá, toda aplicación desarrollada por los clientes basada en los mismos principios y tecnologías también pertenece a este grupo.
- **Oracle Fusion Middleware.** Oracle Fusion Aplications son generalmente aplicaciones web, y por ello necesitan un servidor de aplicaciones y unas infraestructuras. Oracle Fusion Middleware es el término utilizado para describir a la plataforma sobre la que se ejecutan las aplicaciones de Oracle Fusion.
- **Oracle Fusion Architecture.** Este término es el menos conocido, y es el conjunto de buenas prácticas y tecnologías utilizadas para desarrollar aplicaciones Fusion sobre Fusion Middleware. Esto incluye SOA, Java Platform, Java EE y otros.

Oracle eligió para Fusion Java EE, SOA y Web 2.0 como los pilares tecnológicos fundamentales, pero alrededor de estos pilares están incluidos un enorme número de tecnologías y estándares (JPA, EJB, JDBC, BPEL, XML, SOAP, y esto por nombrar solo unos pocos).

Para los desarrolladores esto es un gran problema, ya que ellos deben conocer

todas estas tecnología de abajo a hasta arriba, y con este elevado número, solo está al alcance de los más talentosos. Además para los desarrolladores que estén acostumbrados al enfoque declarativo de programación, utilizado hasta ese momento para desarrollar en Oracle E-Business Suite, es un paso atrás, ya que en vez de centrarse en desarrollar código para resolver las necesidades del negocio tienen que enfrentarse a un enorme número de tecnologías que deben ser entendidas, integradas y mantenidas.

La solución de Oracle a este enorme problema fue Oracle ADF framework y Oracle JDeveloper.

2.2. ADF Framework

Como ya se ha dicho antes Oracle ADF es un framework que simplifica el desarrollo de aplicaciones con Java, Java EE y SOA, y fue desarrollado específicamente para cubrir el requisito de unir todo el gran número de subframeworks y estándares creando un completo framework basado en metadatos para construir aplicaciones empresariales.

Aunque Oracle ADF y JDeveloper son dos productos separados que pueden ser usados independientemente, su máximo potencial se consigue cuando se utilizan juntos. Oracle ADF provee un conjunto de distintas características que pueden ser fácilmente creadas y configuradas a través del JDeveloper.

Oracle ADF está basado alrededor del principio de Modelo-Vista-Controlador (MVC), lo que significa que el framework separa la implementación de los servicios de negocio (Modelo) de la implementación de la interfaz de usuario (Vista), con el Controlador manejando el flujo de la aplicación.

Oracle ADF implementa MVC y además separa la capa de modelo de los servicios de negocio para permitir un desarrollo de aplicaciones basado en servicio. La arquitectura de ADF está basada en cuatro capas:

- **Capa de servicios de negocio.** Provee acceso a los datos desde distintas fuentes y maneja la lógica de negocio.
- **Capa de modelo.** Provee una capa de abstracción sobre la capa de servicios de negocio, permitiendo que las capas de vista y controlador trabajen con diferentes implementaciones de servicios de negocio de una manera consistente.
- **Capa de controlador.** Provee un mecanismo de control de flujo para la aplicación web.
- **Capa de vista.** Provee la interfaz de usuario de la aplicación.

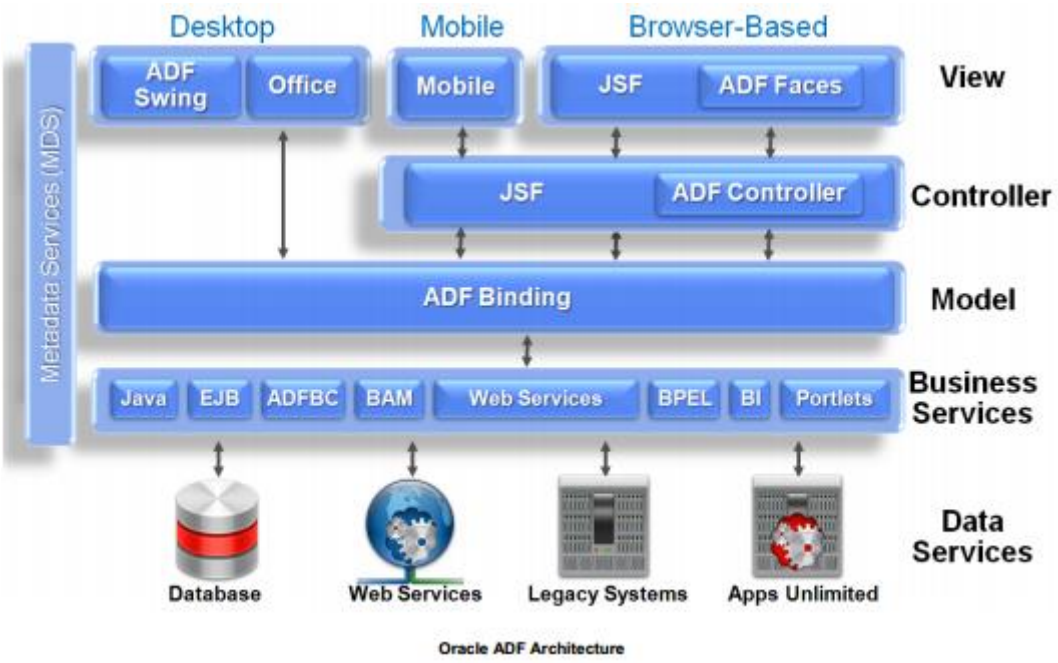

Ilustración 11. Arquitectura de ADF

Oracle ADF permite a los desarrolladores elegir la tecnología que prefieran usar cuando están implementando una determinada capa. En la ilustración se puede observar las distintas opciones disponibles para los desarrolladores a la hora de construir una aplicación ADF. El nexo que integra los distintos componentes de aplicaciones Java EE y permite un desarrollo tan flexible es la capa de modelo.

2.2.1. Servicios de negocio

La capa de servicios de negocio maneja la interacción con la capa de persistencia. Provee servicios como persistencia de los datos, mapeo objeto-relacional, control de transacciones y la ejecución de la lógica de negocio.

La capa de servicios de negocio puede ser implementada en cualquiera de las siguientes opciones:

- Clases Java
- EJB
- Web Services
- JPA
- Oracle ADF Business Components

Los datos además se pueden obtener de diferentes orígenes, base de datos, web services o incluso directamente de archivos (XML o CSV) y de REST.

2.2.1.1. ADF Business Components

ADF Business Components (ADF BC) es la implementación propia de Oracle ADF framework de la capa de persistencia de los servicios de negocio, y por tanto para los desarrolladores que están familiarizados con lenguajes como SQL y PL/SQL es la mejor opción (y para los demás también ya que ADF y JDeveloper están desarrollados para proporcionar un soporte perfecto para ADF BC). Con ADF BC se evita a los desarrolladores que deban escribir el código requerido para:

- Conectar a la base de datos
- Recuperar datos
- Bloquear filas de la base de datos
- Manejar transacciones

Lo más importante es que ADF Business Components ahorra tiempo a los desarrolladores ya que con JDeveloper el uso es totalmente de manera automática y declarativa, haciendo el desarrollo de servicios de negocio más productivo.

ADF BC proporciona una base de ligeras clases Java y ficheros de configuración de metadatos XML que permite a los componentes de la capa de negocio que aprovechen las siguientes funcionalidades:

Simplificar el acceso a datos.

ADF Business Components mejora la manipulación de los datos ya que permite diseñar un modelo de datos para mostrar a los clientes con sólo la información necesaria, o incluir las jerarquías maestro-detalle que se puedan dar.

También automatiza la coordinación entre los cambios de la capa de modelo con la capa de servicios de negocio, al igual que valida y guarda automáticamente cualquier cambio en la base de datos.

Reforzar la validación de datos y la lógica de negocio.

De forma declarativa se puede establecer campos obligatorios, campos únicos o de clave primaria, la precisión y las claves foráneas, e incluso reglas complejas que pueden ser realizadas programática o declarativamente.

Soportar UIs con múltiples unidades de trabajo.

ADF BC refleja automáticamente en la interfaz de usuario los cambios que se realicen en la lógica de los servicios de negocio y además abstrae a los desarrolladores de tareas como manejar imágenes, sonidos o videos, gestionar

los mensajes de error, las máscaras de los campos, tooltips y otros.

En definitiva ayuda a manejar la información que es utilizada en la interfaz de manera coherente y automatizada, evitando al desarrollador la ardua tarea de escribir el código que fuera necesario.

Implementar arquitectura SOA.

Se pueden crear interfaces de servicios web para la integración del negocio, sin escribir una sola línea de código y siempre basado en las mejores prácticas de programación.

Además se simplifica la seguridad de la aplicación haciendo uso de JAAS.

Facilidad de customización.

Una vez finalizado el desarrollo de la aplicación, se pueden extender nuevas funcionalidades sin tener que modificar el código fuente ya desplegado, añadiendo nuevos componentes o sustituyendo por otros con funcionalidades ampliadas.

ADF Business Components implementa el servicio de negocio a través del siguiente set de componentes:

- **Application Module.** Un Application Module es un contenedor para View Objects, View Links y otras instancias del módulo de aplicación. Encapsula el modelo de datos de los ADF BC. En una aplicación siempre ha de haber un módulo de aplicación raíz que provea el contexto de transacciones.
- **Entity Object.** Representa una tabla en la base de datos y una instancia de una fila de esa tabla. ADF Entity Object encapsula datos de la aplicación, reglas de validación de los campos que contiene y persistencia.
  Como ya se ha comentado se permite la opción de desarrollar ficheros XML de metadatos, y los Entity Objects no son una excepción, cuando se crea un nuevo Entity Object también se crea el fichero XML de metadatos que recoge su definición. La definición de un Entity Object describe la estructura y actúa como plantilla para crear la instancia durante la ejecución de la aplicación.
- **View Object.** El rol principal de un View Object es proveer a la interfaz de usuario los datos que los componentes van a utilizar.
  Los View Objects pueden ser de escritura o de solo lectura. Un View

Object será de escritura si está basado en uno o varios Entity Objects, y guardará todas las operaciones que se realicen sobre él a través del Entity Object. Por otro lado será de solo escritura si no está basado en un Entity Object, esto sucede si representa los datos de una consulta SQL o de una lista estática.

Al igual que los Entity Object tienen su fichero de metadatos XML en el que se define su estructura, y pueden configurarse para cambiar el ciclo de vida de la recuperación de datos durante la ejecución.

- **Association.** Una asociación describe la relación que existe entre dos Entity Objects, cuando se crea una se establece un enlace accesible desde el código Java para trabajar con el Entity dependiente. El enlace puede crearse de una clave foránea de la base de datos o manualmente cuando se crea.
- **View Link.** Un View Link como una asociación describe una relación, pero en este caso entre dos View Objects. Se puede crear manualmente como una asociación, o puede crearse directamente de una asociación ya existente.

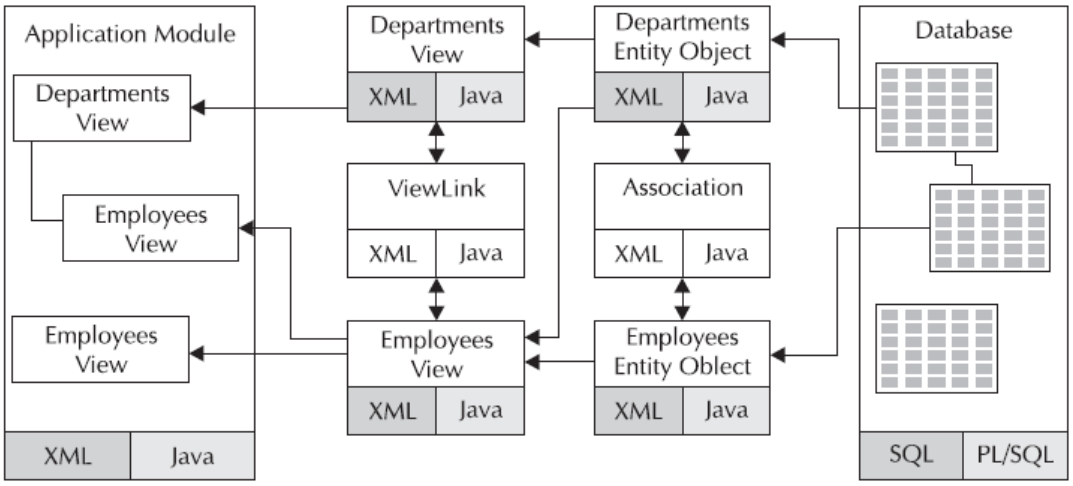

Ilustración 12. Visión global ADF Bussiness Components

2.2.2. Controlador

La capa de controlador maneja el flujo de la aplicación y los parámetros de entrada del usuario. Por ejemplo, si se hace click sobre un botón en la página, el controlador determina que acción se realiza y a donde hay que navegar.

Para implementar esta capa ADF nos proporciona dos opciones, controlador JSF estándar y controlador ADF (que extiende la funcionalidad directamente del controlador estándar de JSF). Sea cual sea la opción elegida el diseño del flujo de la

aplicación se realizará asignando layouts a las páginas y reglas de navegación en un diagrama.

Con el controlador de ADF puedes separar el flujo de la aplicación en flujos de tarea más pequeños ("*task flows"*). En los "*task flows*" se incluyen componentes no visuales tales como llamadas a métodos o puntos de decisión del flujo, y se pueden crear además como "*page fragment flows"* que se ejecutan dentro de una región de una página. Este enfoque maximiza la reutilización y la simplificación de la integración en portales y aplicaciones web híbridas.

2.2.2.1. ADF Controller

El motor de ADF Controller está definido en la clase "*NavigationHandlerImpl*", que extiende de "*javax.faces.application.NavigationHandler"* de JSF. Se crea el archivo "*faces-config.xml"* en el directorio "*META-INF"* de ADF Controller, que registra el controlador ADF con el de JSF durante la ejecución.

Una vez el registro se ha realizado con éxito, ADF Controller maneja todas peticiónes, es decir, JSF delega toda la navegación en ADF. Aun así si una petición no puede ser controlada por el controlador ADF, se pasa el control al manejador de JSF (ya que ADF Controller está desarrollado por encima de JSF).

El controlador de ADF además tiene una seguridad integrada que obliga al control de acceso en las páginas ligadas a los Task Flows de ADF (en el caso en el que Oracle ADF Security haya sido configurado) por medio de "*Java Authentication and Authorization Service (JAAS)"*.

2.2.2.2. Task Flows

El flujo de una aplicación generalmente envuelve la navegación entre las distintas páginas, pero ADF va más allá de un flujo de páginas añadiendo llamadas a métodos de servicio y características como condicionamiento de la ruta del flujo. Cada una de estas definiciones del flujo de aplicación es llamada Task Flow y son muy fácilmente reutilizables.

Cada Task Flow se define en su propio archivo, así el flujo de la aplicación puede ser separado en diferentes Task Flows. A parte de la reutilización los Task Flows tienen más beneficios, como realizar pruebas más concretas o independencia en el desarrollo, pero el más importante es que facilita el entendimiento de la aplicación.

Hay dos tipos de Task Flows:

Unbounded Task Flows

Se puede concebir un Unbounded Task Flow como Task Flow raíz desde el que la aplicación es ejecutada. En un Unbounded Task Flow, todas las páginas que están contenidas en él pueden ser la página de inicio de la aplicación, por ello si solamente quieres tener una, solo debes tener una página definida en el Task Flow.

Generalmente un Unbounded Task Flows tiene todas las páginas que pueden lanzar la aplicación y esas páginas tienen referencias a otros Task Flows.

Bounded Task Flows

Este es el tipo de Task Flows con los que se trabaja más a menudo, y representan el flujo de aplicación que puede ser referenciado desde cualquier otro Task Flow (bounded o unbounded).

Un Bounded Task Flows tiene un único punto de entrada y cero, uno o varios puntos de salida (se pueden entender parecidos a una subrutina de un lenguaje de programación, se puede llamar desde distintos sitios pero solo tiene un punto de entrada, puede ser parametrizada y tienen múltiples puntos de retorno dentro de su código de ejecución).

Un caso particular de los Bounded Task Flows son las ADF Regions. En las aplicaciones JSF la navegación debía ser externa a la página actual para completar una subtarea del proceso de negocio. Una Task Flow definida como Bounded puede ser creada como una región de ADF Faces que permite la navegación a través de distintas tareas sin tener que salir de la página.

2.2.3. Vista

La capa de la vista representa la interfaz de usuario de la aplicación.

Oracle ADF soporta acceso múltiple a los servicios de negocio permitiendo la reutilización de los servicios y acceder a ellos desde el cliente web, aplicaciones de escritorio, Microsoft Excel o dispositivos móviles como smartphones.

Para la interfaz web Oracle ADF ofrece un conjunto de 150 componentes JSF (soportan Ajax) que simplifican la creación de la dinamicidad y apariencia de las interfaces de usuario.

2.2.3.1. ADF Faces

Como ya se ha explicado JSF es un framework J2EE estándar para construir interfaces de usuario en aplicaciones web, y su principal ventaja es que cada componente gestiona su comportamiento y como se dibuja en la interfaz, es decir, el desarrollador se abstrae de tareas como hacer scroll en un componente, o como se genera el código HTML del componente dependiendo del navegador y del dispositivo.

Como JSF es un estándar, cada compañía puede desarrollar sus propios componentes JSF. ADF Faces son los componentes JSF creados por Oracle integrados en ADF framework.

Además de los componentes UI, ADF Faces es mucho más que eso, tiene diferentes bloques:

Componentes UI

ADF Faces cuenta con una paleta con 150 componentes que engloban desde simples botones y campos de entrada hasta gráficos o mapas. Además de componentes visuales también ofrece componentes que añaden "*drag and drop*", escuchadores de eventos en el cliente y otros muchos.

Cada componente tiene unas propiedades y comportamientos propios, por ejemplo un botón puede iniciar una acción cuando es pulsado o un campo de entrada cuando se hace foco en él.

Backing Beans

Aunque la lógica de la aplicación debe estar contenida en los servicios de negocio, puede ocurrir el caso en el que se necesite escribir código para personalizar la interfaz de usuario. En JSF existen los "*Managed Beans*" que son clases java gestionadas por JSF donde se puede escribir código especifico de la interfaz.

Backing Bean es el nombre que se ha establecido para llamar a los "*Managed Beans*" que se encargan únicamente de la interfaz de usuario.

Expression Language

Expression Language (EL) son simples anotaciones que permiten a un componente UI referirse a un Java Bean asociado de la aplicación.

Ciclo de vida

Cuando se hace una petición o se realiza una acción en una página se realizan una serie de pasos. Estos pasos son el ciclo de vida, y la mayor parte no son necesarios para el desarrollador, pero aun así es bueno saber cuáles son y según se gana experiencia también como aumentar el comportamiento y funcionalidad de estos pasos.

2.2.4. Modelo

La capa de modelo conecta los servicios de negocio con los objetos de otras capas que los usan. Proporciona una implementación de la capa de modelo que se asienta sobre los servicios de negocio proveyendo una única interfaz para acceder a cualquier tipo de servicio de negocio. La capa de modelo está compuesta por dos componentes, "*data controls*" y "*data binding*"

2.2.4.1. Data Controls

Crean una abstracción de los detalles de implementación de los servicios de negocio a los clientes.

Estos son creados o modificados automáticamente, en un fichero separado (.dcx) en el que se almacena su descripción, cuando se añaden View Objects al Applicacion Module. Los Data Controls son mostrados en la paleta de Data Controls que proporciona JDeveloper, en la que se muestran los atributos, métodos, operaciones tanto de cliente como de módulo de aplicación.

2.2.4.2. Data Bindings

Proporciona acceso a los métodos y atributos de los "*Data Controls"* a los componentes de la vista, creando una separación clara de la vista y el modelo. Los Data Bindings son creados en un archivo XML separado, y cada página de la vista tiene un archivo de Data Bindings propio.

Hay 3 tipos de Data Bindings:

Executable Bindings

Son los "*Iterator Bindings*", que simplifican la construcción de interfaces que permiten paginar y hacer scroll sobre colecciones de datos. También incluyen bindings que permiten hacer búsquedas y anidas una serie de páginas dentro de otra.

Value Bindings

Son los bindings usados por los componentes de la interfaz de usuario para mostrar los datos. Pueden ser desde muy sencillos como un simple campo de un View Object, hasta un binding de tipo árbol en el que se tienen datos estructurados jerárquicamente en base a más de un View Object.

Action Bindings

Usados por componentes como hipervínculos o botones para ejecutar operaciones básicas o personalizadas sobre los data controls.

Debido a la arquitectura de metadatos de la capa de modelo, los desarrolladores obtienen una misma experiencia cuando se liga cualquier tipo de servicio de negocio con las capas de vista y controlador.

2.3. JDeveloper

Para realizar cualquier tarea es importante tener la correcta herramienta que facilite la realización del trabajo. La herramienta para el desarrollo *"Fusion Applications"* es Oracle JDeveloper y está creada precisamente para desarrollar aplicaciones con el framework ADF.

JDeveloper provee e integra un entorno de desarrollo (IDE) que ayuda a los desarrolladores en todo el ciclo de vida de crear *"Fusion Applications".*

JDeveloper proporciona buenas herramientas para el desarrollo en Java, y ayuda en las tareas de escribir, construir y ejecutar programas Java. Además provee características como chequeo de sintaxis, refactorización y formateo de código,

plantillas, comparación de archivos, debugueadores y otras más. Aun así JDeveloper no es solo Java, va mucho más allá.

Además contiene herramientas para diseñar las páginas, un inspector de propiedades o paletas de componentes UI para construir páginas de la interfaz de usuario con la simple acción de arrastrar un componente en un determinado punto de la página y ajustando sus propiedades en el inspector de propiedades. Además el desarrollo de servicios SOA se lleva a cabo a través de "*wizards*" por lo que no debes escribir ficheros de configuración.

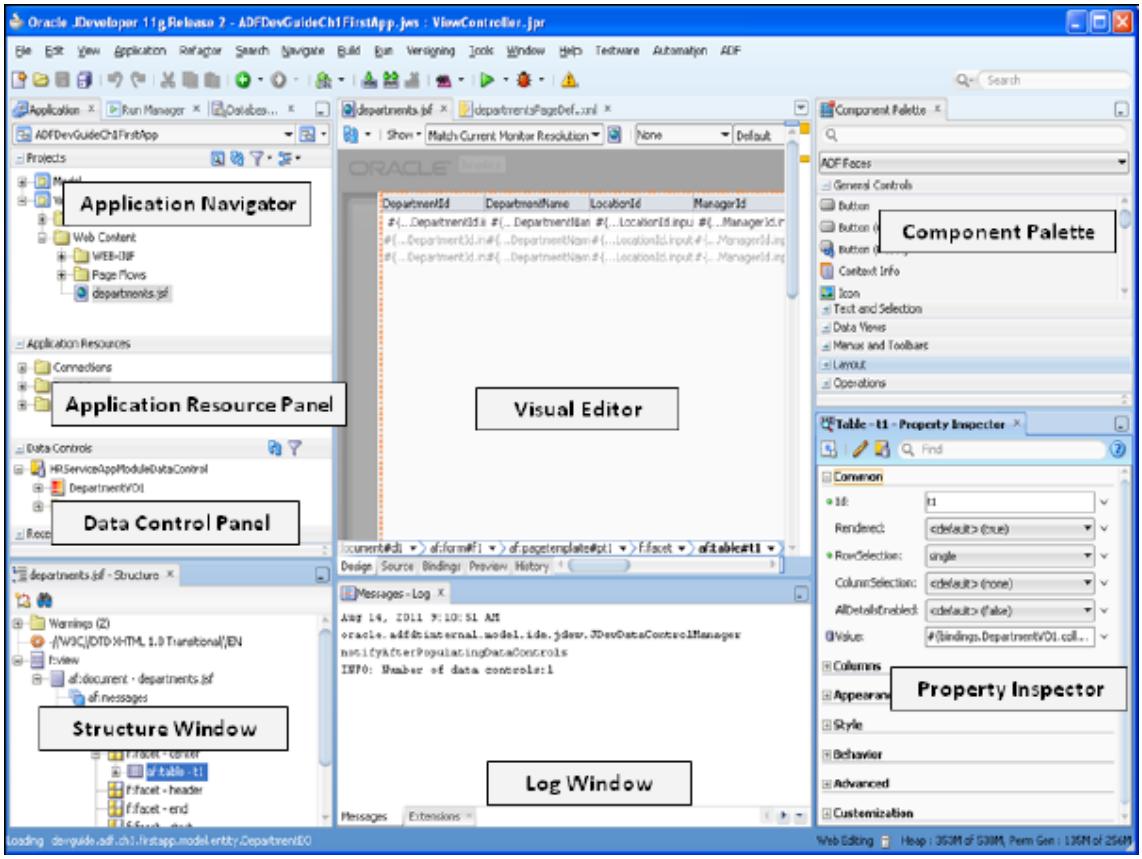

Ilustración 13. Visión Global JDeveloper

- **Application navigator:** ayuda a manejar los contenidos y recursos asociados de una aplicación. Puedes crear nuevos proyectos y archivos usando las herramientas de esta ventana.
- **Application resource panel:** muestra los recursos a nivel de aplicación y los ficheros de configuración, esto incluye la información de conexión a la base de datos, los archivos de metadatos de usados para configurar los ADF Bussiness Components, y muchos otros.

- **Data Control panel:** muestra las colecciones de datos, de atributos, operaciones y métodos de los servicios de negocio que han sido utilizaos. Los distintos objetos que se muestran pueden ser arrastrados en la interfaz de usuario, lo que generara los archivos de metadatos en XML necesarios para ligar los servicios de negocio con la UI.
- **Structure window:** muestra una vista estructurada del contenido de los componentes que están seleccionados en el "Visual Editor".
- **Visual Editor:** el editor visual ayuda a desarrollar de una manera visual la interfaz de usuario. Permite HTML, JSP, JSF, Facelets, UI de aplicaciones móviles y Java Swing.
- **Component palette:** la paleta de componentes lista todos los componentes disponibles asociados a la tecnología seleccionada para diseñar las páginas.
- **Property inspector:** el inspector de propiedades muestra todas las propiedades del componente seleccionado, ayudando al desarrollador a cambiarlas de una manera muy sencilla y práctica.
- **Log window:** muestra los distintos logs que intervienen (compilador, debugger, etc…)

2.4. Weblogic

Oracle Weblogic es un servidor de aplicaciones J2EE desarrollado por BEA Systems y que tras la adquisición de éste por parte de Oracle se ha convertido en su servidor principal de aplicaciones Java. Es importante resaltar el objetivo para el que Weblogic está diseñado, "*grid computing*". Grid Computing permite a grupos de ordenadores conectados por red estar agrupados y preparados para satisfacer los cambios necesarios del negocio. En vez de servidores dedicados y almacenamiento para cada aplicación, grid computing permite compartir a múltiples aplicaciones la misma infraestructura, proveyendo mayor flexibilidad, potencia, rendimiento, escalabilidad y sobre todo disponibilidad.

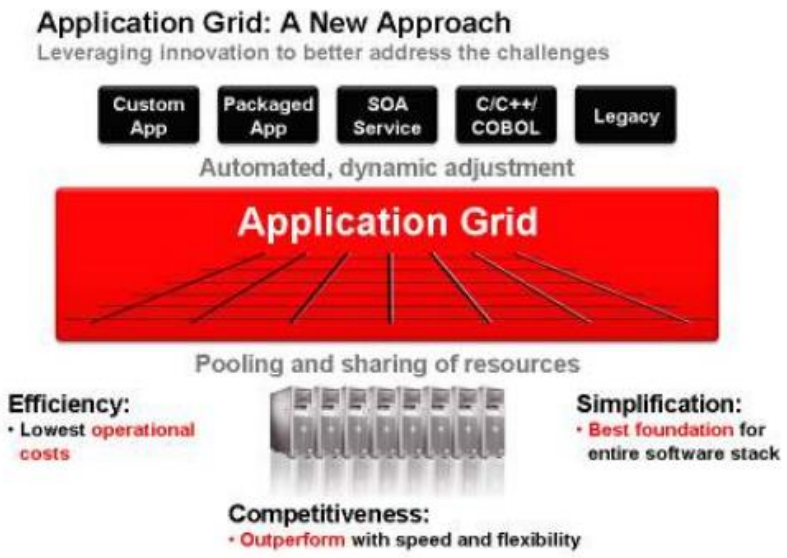

Ilustración 14. Vision global de Weblogic

En la actualidad Oracle Weblogic es el servidor que proporciona mejor rendimiento, tanto si las aplicaciones se ejecutan en una como en varias instancias del servidor. Esto se debe en gran medida a Oracle JRockit, que es la máquina virtual de java más rápida del mercado. Oracle JRockit es la interfaz entre el servidor de aplicaciones y el hardware, y está optimizada para sistemas de 32 y 64 bits.

Con lo expuesto se pude entender perfectamente el porqué de la integración de Weblogic como servidor principal de Oracle Fusion.

3. Conceptos básicos en el desarrollo de una nueva aplicación con Oracle ADF framework y Oracle JDeveloper

3.1. Primeros pasos crear una aplicación

Aunque JDeveloper ha sido diseñado principalmente para desarrollar aplicaciones fusion, también se pueden realizar otras muchas tareas , por ello el primer paso en la creación de una nueva aplicación es elegir el tipo de aplicación que se quiere crear.

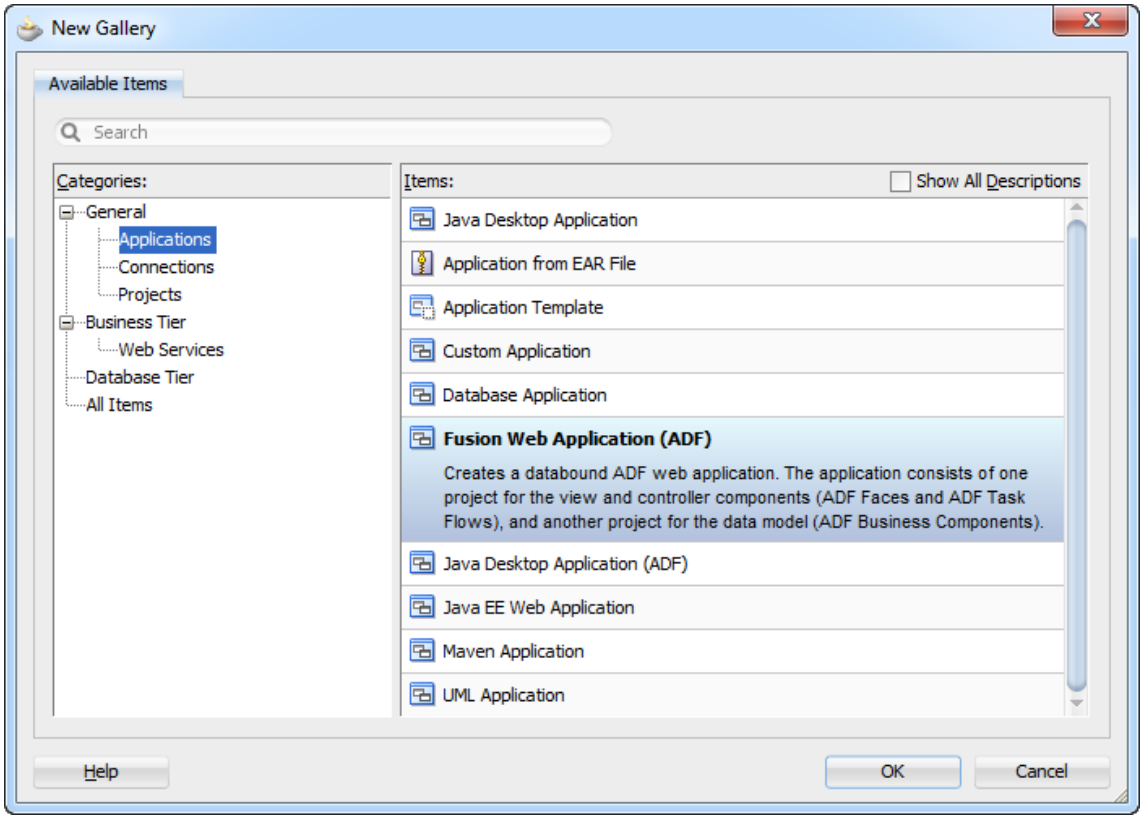

Ilustración 15. Crear nueva aplicación fusión

Una vez seleccionada la opción Fusion Web Applicaction (ADF), el JDeveloper a través de un "wizard" permitirá establecer el nombre de la aplicación y el directorio en el que guardar el proyecto.

Los siguientes pasos del "wizard" configuraran el esqueleto básico de la aplicación a través de generar los proyectos "Model" y "View Controller". Como indica su nombre estos proyectos contendrán los servicios de negocio y la UI relacionada respectivamente.

Una vez terminado el “wizard” se creara la nueva aplicación con la siguiente estructura.

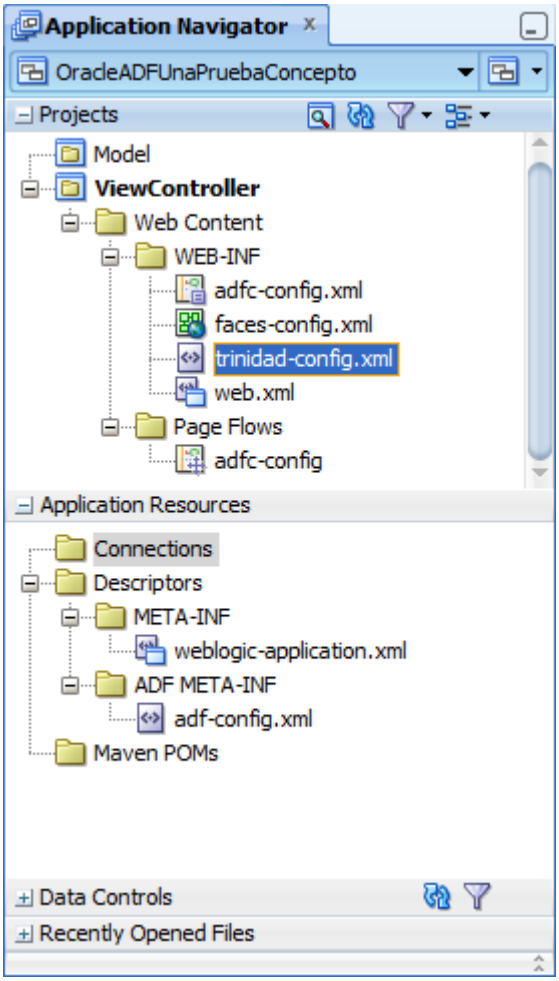

Ilustración 16. Estructura de un proyecto fusión

- **adfc-config.xml:** ADF Faces extiende la funcionalidad de JSF, y utiliza adfc-config.xml para mantener las configuraciones. En él se configura las definiciones de los “managed beans” utilizados, los casos de navegación y muchos otros.
- **faces-config.xml:** contiene las configuraciones de una aplicación web construida usando JSF. Permite configurar los “managed beans”, conversores de datos, validadores usados en la UI, navegación, etc…
- **trinidad-config.xml:** la base del conjunto de componentes de ADF Faces está formada por Apache MyFaces Trinidad (de hecho los componentes de Trinidad antes pertenecían a ADF Faces pero fueron donados). Por defecto contiene únicamente el nombre de la “skin-family”, pero

además se pueden cambiar las configuraciones por defecto de accesibilidad, administración de estados y otras.

- **web.xml:** este fichero actúa como el descriptor de despliegue de una aplicación J2EE y es generado automáticamente por el JDeveloper. Se pueden configurar parámetros de contexto y distintos filtros. Además incluye escuchadores de contexto a nivel de servlet para inicializar la administración y monitorización de servicios de las capa de vista y modelo.
- **weblogic-application.xml:** es el descriptor de despliegue específico para el servidor Weblogic.
- **adf-config.xml:** contiene configuraciones a nivel de aplicación que administran el modo de ejecución, tales como error en los módulos de aplicación, número límite de filas para los View Object, "PPR" automático y otras.

3.2. ADF Business Services

ADF soporta diferentes orígenes para los servicios de negocio en función de las necesidades de la aplicación que se va a desarrollar y aunque en la mayoría de los casos se trate de una base de datos, el desarrollador tiene la opción de utilizar otras alternativas que satisfagan los requisitos.

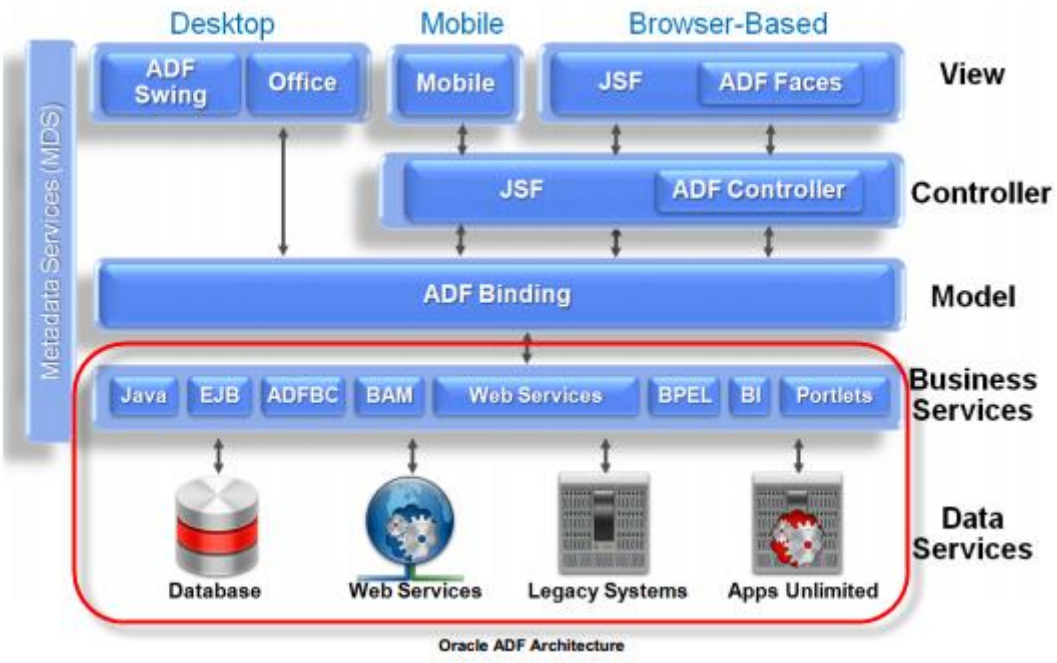

Ilustración 17. Distintas opciones de los servicios de negocio

Además de los distintos orígenes de datos para los servicios que ADF soporta, también permite utilizar diferentes tecnologías para implementar los servicios de negocio independientemente del origen de los datos de la aplicación, aunque la más recomendada es ADF Bussiness Components debido a que no requiere al desarrollador escribir código para crear la infraestructura de la aplicación.

3.2.1. Base de datos

JDeveloper permite la creación una conexión con una base de datos de forma muy sencilla a través de un "*wizard*". La conexión puede ser realizada a distintos tipos de bases de datos y además permite la configuración de la conexión de diferentes formas.

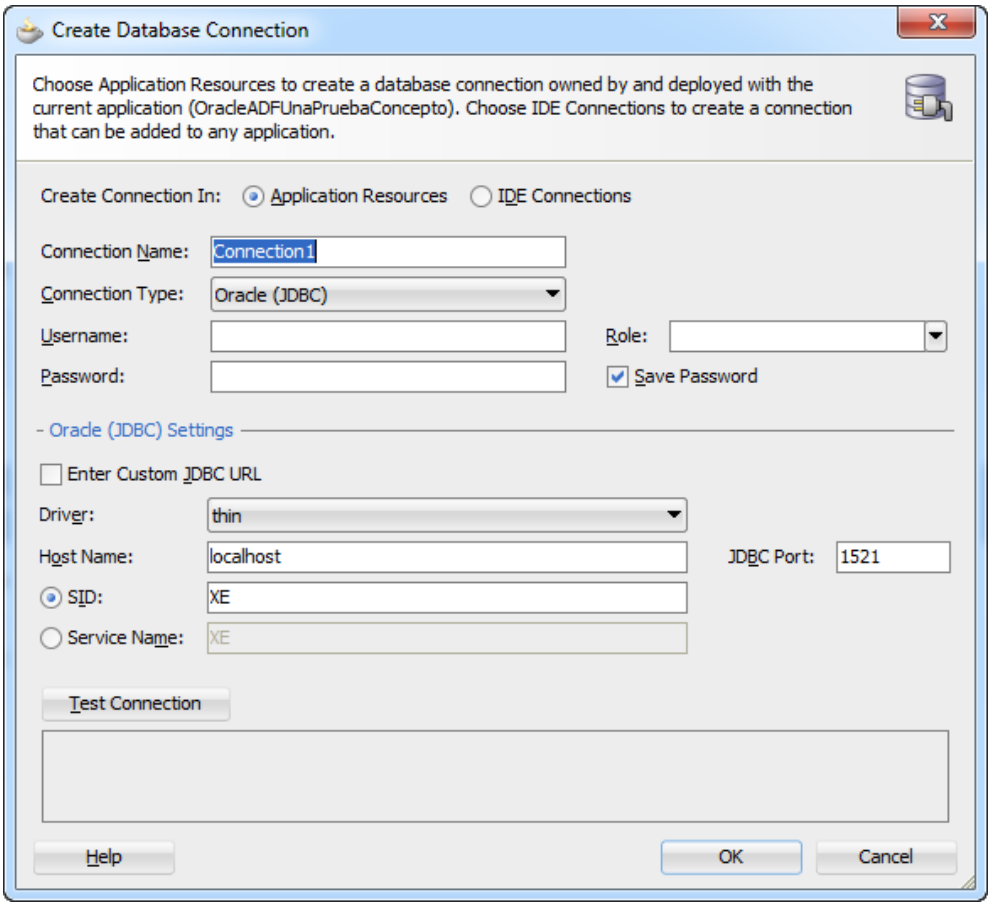

Ilustración 18. Creación de una nueva conexión de base de datos

Como se puede observar a parte configurar los parámetros básicos para una conexión de una manera visual para los distintos tipos de base de datos, existe la

opción de introducir una cadena JDBC para crear la conexión.

Tras configurar la conexión con la base de datos el JDeveloper genera una serie de archivos de metadatos XML que mantienen la información de la conexión.

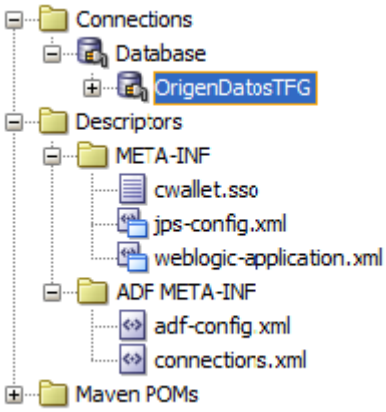

Ilustración 19. Ficheros nueva conexión

- **cwallet.sso:** en este archivo se guardan los credenciales para conectar a sistemas externos y sigue la especificación "*Oracle Platform Security(OPS)*"
- **jps-config.xml:** este archivo se usa para almacenar las configuraciones OPS, y la localización de este archivo se guarda en adf-config.xml. Si no está configurada la seguridad de ADF este archivo actúa como un simple puntero a cwallet.sso.
- **connections.xml:** en este fichero se guardan todos los detalles de las conexiones de la aplicación, tanto a base de datos como a otros tipos de conexión.

3.2.2. Web Service

Existen diferentes formas de diseñar una aplicación ADF para hacer uso de servicios web, aunque se puede crear un proxy al web service que se quiere utilizar no es la opción más recomendable ya que la llamada a los web services se realizaría en la capa de vista y controlador desde un Managed Bean. La opción más recomendable según "*ADF Best Practices*" es crear los componentes de los servicios de negocio directamente usando la interfaz del web service.

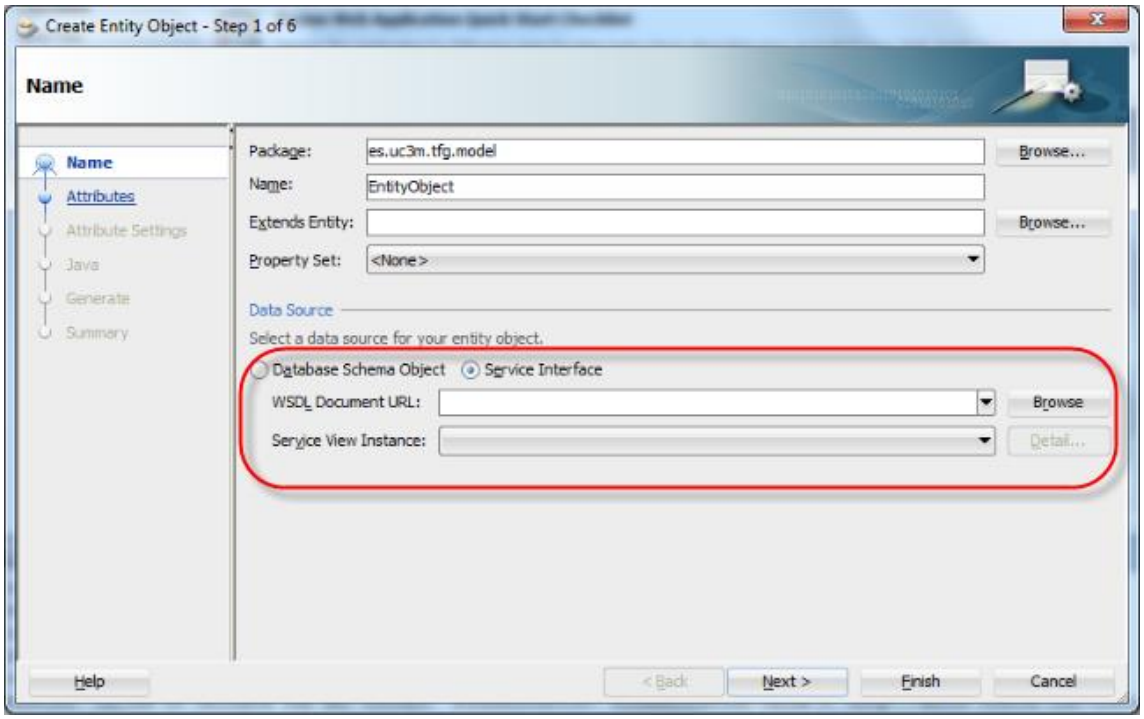

Ilustración 20. Ejemplo creación Entity Object desde Web Service

3.2.3. Creación ADF Bussiness Components

Cuando se desarrollan los ADF BC, JDeveloper consulta a la base de datos y lee la información sobre las tablas sobre las cuales el servicio de negocio va a estar basado. Por ejemplo, las columnas de una determinada tabla, los tipos de estas, su precisión o si el campo puede ser nulo e incluso las claves foráneas. Toda esta información es codificada en archivos XML que forman parte de la aplicación.

Si se desea cambiar algo de esta información, como el orden de los datos o si un atributo es actualizable, únicamente ha de irse a los editores de ADF Bussiness Components y cambiar las propiedades necesarias.

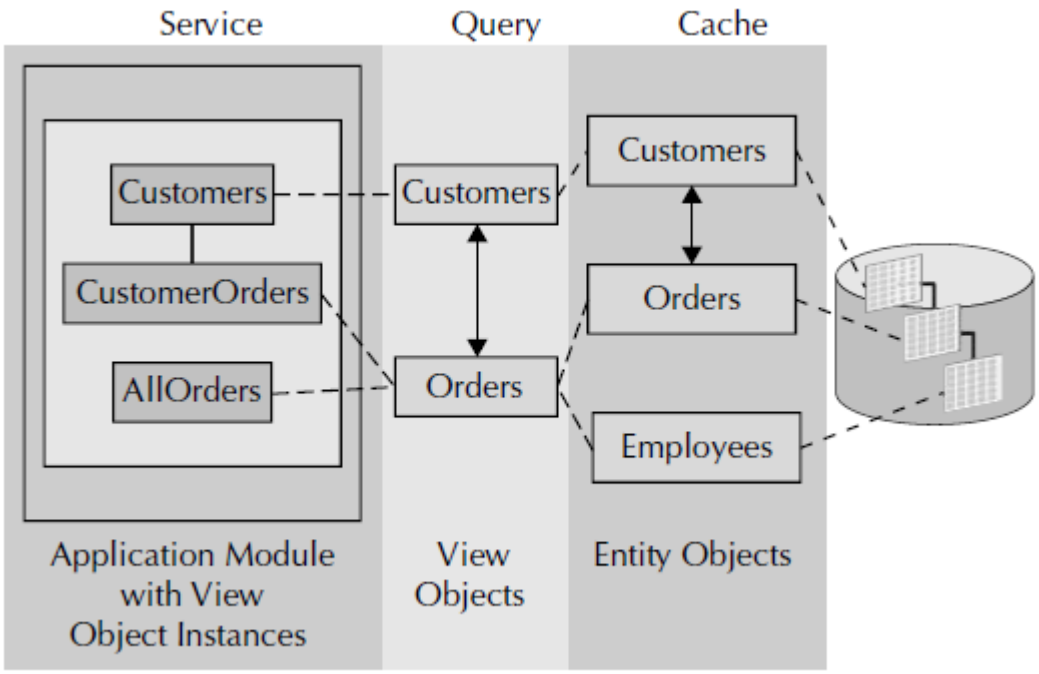

Ilustración 21. Relaciones entre los distintos ADF Bussiness Components

Los tres bloques más importantes son los Entity Objects, View Objects y el Application Module.

3.2.3.1. Entity Objects

Cuando se construye una aplicación que utiliza una tabla de la base de datos, la aplicación necesita un espacio donde guardar las filas recuperadas de la base de datos, los Entity Objects son los responsables de ello.

Hay dos formas de crear los Entity Objects, de manera múltiple (generalmente al comenzar una aplicación), en la que se selecciona un conjunto de tablas de la base de datos que se mapean directamente con un Entity Object cada una, y de manera individual, para crear un único Entity Object que puede estar mapeado con una tablas de la base de datos o incluso con un web service como ya se ha dicho antes.

Ambas formas se realizarían de forma parecida a través de un “wizard” y el resultado será el mismo, por ello unicamente se va a mostrar un ejemplo de la segunda.

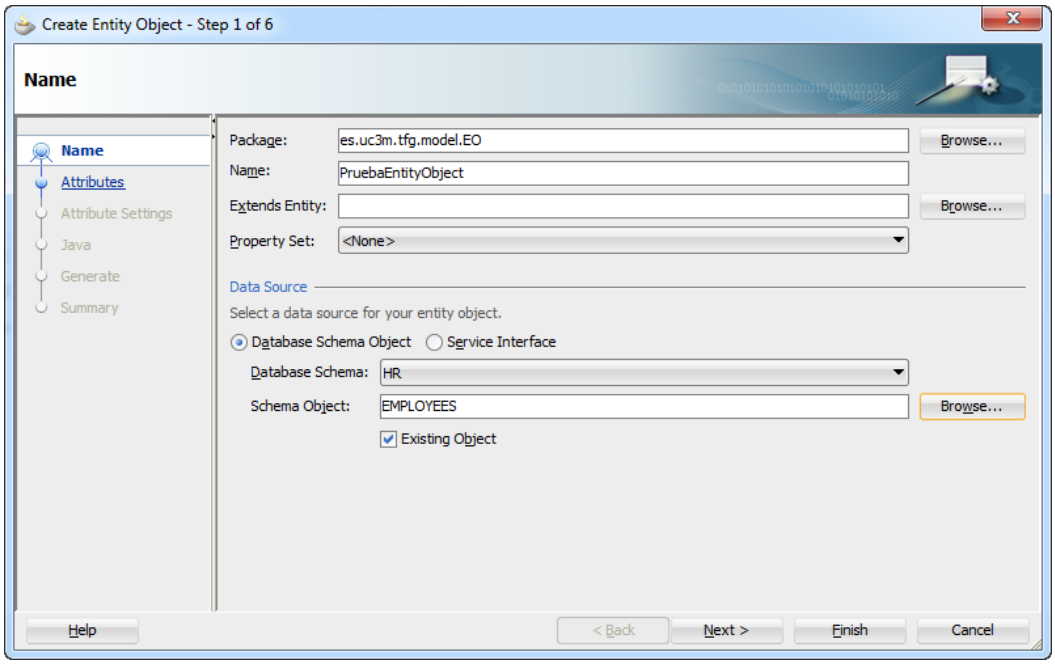

Ilustración 22. Nuevo Entity Object

En el ejemplo se ha creado un nuevo Entity Object con el nombre PruebaEntityObject que tiene como origen de datos la tabla "*Employees*" del esquema HR de la base de datos que se ha configurado antes. Como se puede observar, el JDeveloper consulta la base de datos y muestra todos los posibles esquemas de ésta y posteriormente los objetos de ese esquema, con lo que el desarrollador no necesita conocer el nombre exacto y simplemente tiene que seleccionar el correcto.

En los siguientes pasos del "*wizard*" se seleccionan las columnas de la tabla que quieren ser mapeadas como atributos en el Entity Object y las propiedades de cada atributo.

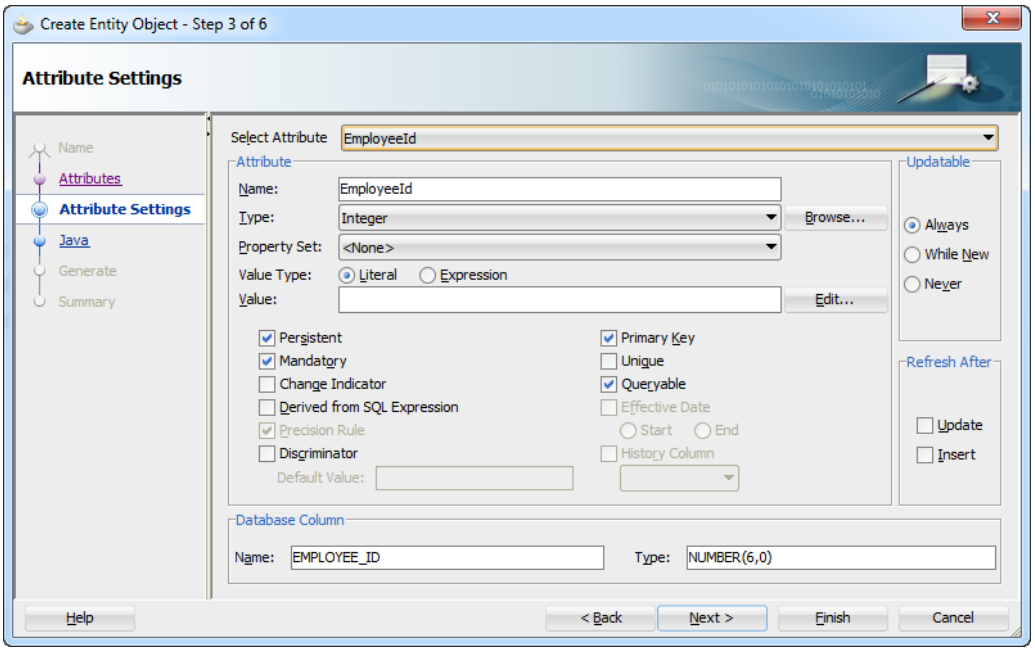

Ilustración 23. Propiedades de los atributos

Una vez se han configurado las propiedades de los atributos se pueden generar las clases Java para extender las funcionalidades por defecto del Entity Object, consiguiendo una customización en función de los requisitos de la aplicación.

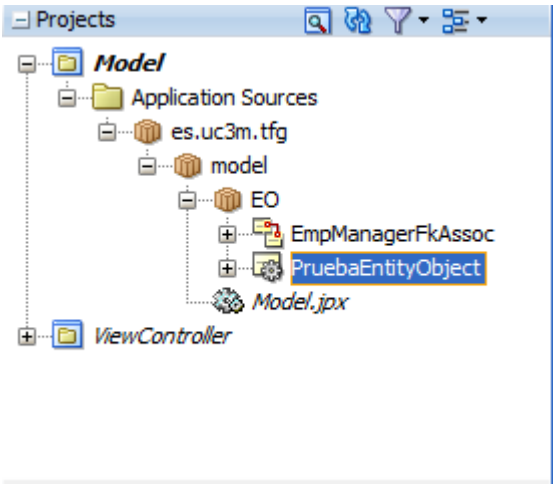

Ilustración 24. Ficheros generados al crear un Entity Object

El resultado de crear un nuevo Entity Object es el que se observa en la ilustración, el fichero PruebaEntityObject (donde se aloja el archivo de metadatos XML y las clases Java del Entity Object si se han creado) y las respectivas asociaciones que

tiene la tabla. En el fichero PruebaEntityObject.xml se pueden realizar todo tipo de configuraciones sobre el Entity Objects tanto de manera declarativa como desde el código fuente.

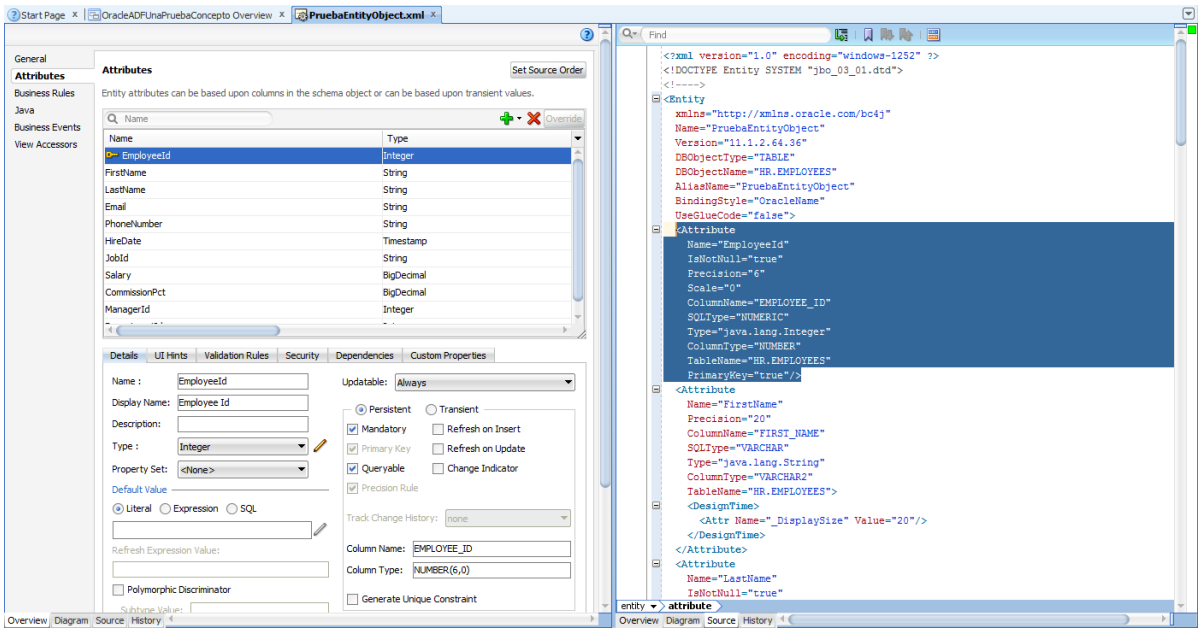

Ilustración 25. Editor de código y visual de un Entity Object

Las relaciones entre distintos Entity Objects se establecen mediante "*Associations*", que son generadas por el "*wizard*" de creación del Entity Object por cada clave foránea que tenga la tabla a la que el Entity Object se refiere. También es posible crearlas manualmente por el desarrollador, y lo más importante es que no es necesario que exista esa relación en la base datos.

3.2.3.2. View Objects

La manera de crear View Objects es similar a la de los Entity Objects, excepto que el rol de los View Objects es dar una forma de acceso a las filas para las diferentes capas, y pueden estar basados en listas estáticas y sentencias SQL (de solo lectura) o en Entity Objects (lectura y escritura). Por ejemplo, si se quiere interactuar con una tabla para insertar nuevos registros, primero se deberá crear el Entity Object asociado a esa tabla y posteriormente un View Object que esté basado en el Entity Object anterior (se le podrán establecer filtros en el caso de que se quieran recuperar un conjunto específico de filas y no todas las filas de la tabla) que se deberá acceder al módulo de aplicación para generar los "*Data Controls*" correspondientes con los que interactuará la vista.

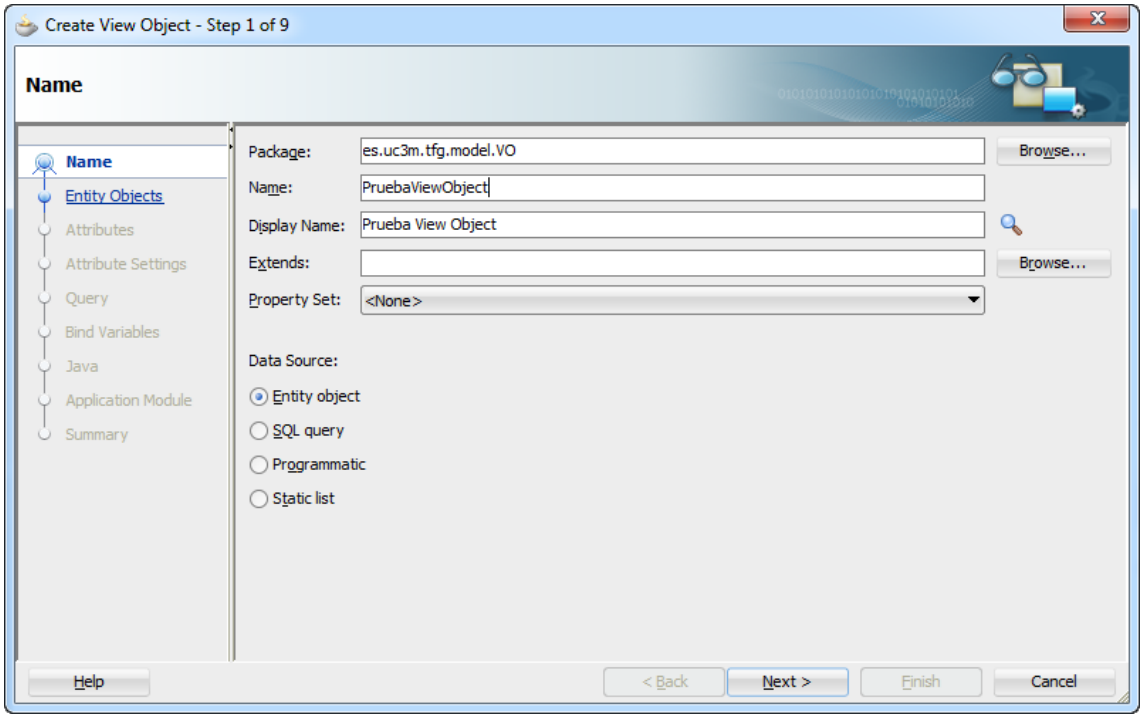

Ilustración 26. Nuevo View Object

Al igual que en la creación de un Entity Object lo primero que se realiza es establecer los nombres del paquete y View Object, y el origen de los datos. Si selecciona Entity Object el origen serán los Entity Objects ya creados en la aplicación, puede ser uno o varios (si son varios hay que establecer las relaciones entre ellos). Al seleccionar las otras opciones los View Objects serán de sólo lectura y los datos recuperados podrán estar en base de datos o ser calculados por el programador.

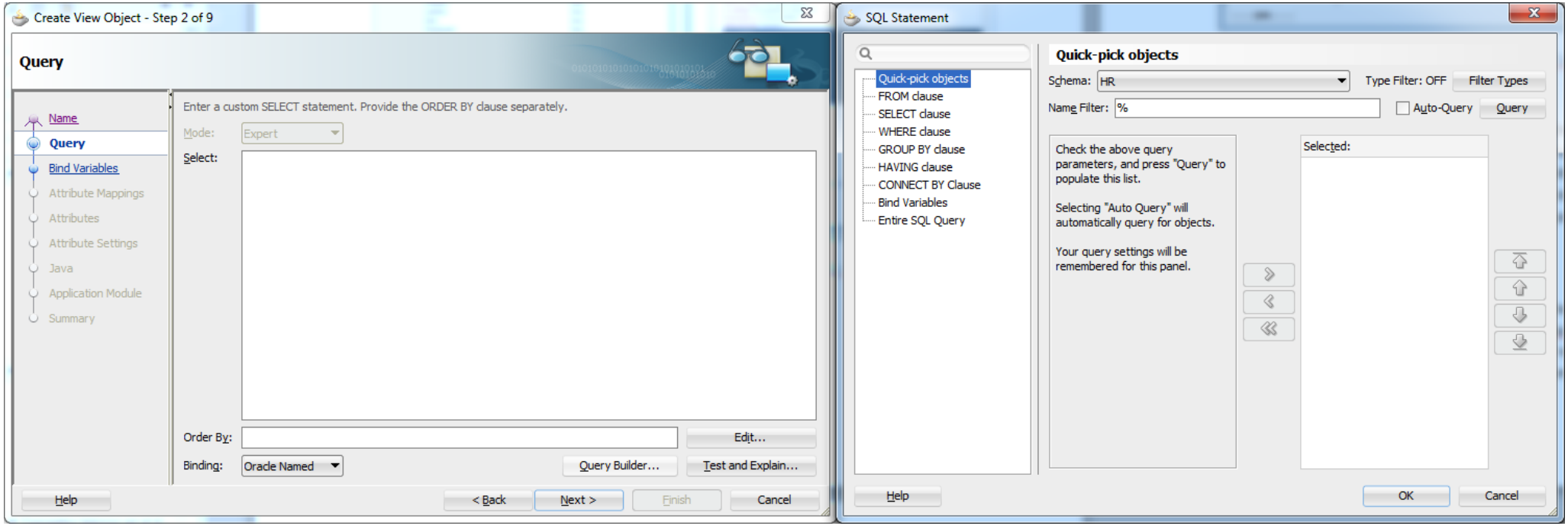

Ilustración 27. Ejemplo crear View Object de una sentencia SQL

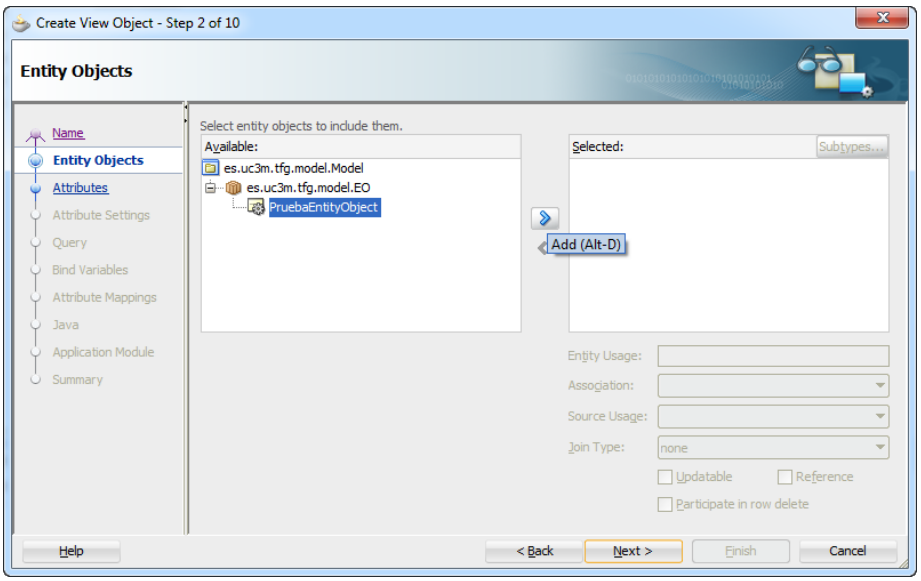

Ilustración 28. Ejemplo crear View Object basado en un Entity Object

En los pasos sucesivos se establecerán los atributos del View Object y sus propiedades, además se pueden establecer variables parametrizadas en el filtro del View Object que se utilizan para recuperar datos de manera dinámica. Por ejemplo, en las listas de valores de un formulario es muy frecuente que dependan de otro campo, por ello supongamos que tenemos un formulario que tiene un campo con una lista de valores de países y otro campo con una lista de valores de provincias. Gracias a esa variable parametrizada en el filtro del View Object se conseguirá que al seleccionar un país en el primer campo, el segundo actualizará su lista de provincias en función del país.

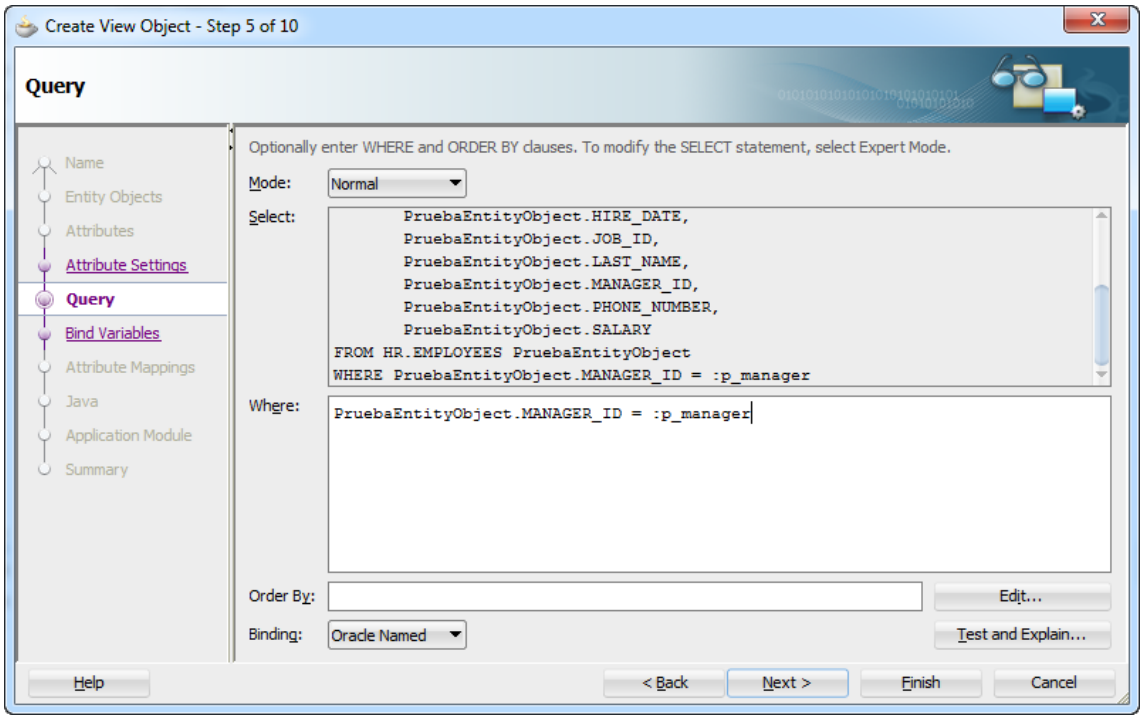

Ilustración 29. Ejemplo crear un filtro en base a una variable parametrizada

Una vez creado el View Object generará un archivo con el nombre que se ha establecido en el "*wizard*" donde al igual que en el Entity Object aloja el fichero de metadatos XML de configuración y las clases Java en el caso de que se quiera customizar el View Object. En el fichero XML de metadatos generado se pueden configurar las propiedades del View Object, como añadir Entity Objects en los que se basa el View Object, cambiar propiedades de los atributos o cambiar la sentencia SQL. Al igual que todos los archivos de metadatos estos cambios pueden realizarse tanto de manera declarativa como escribiendo código XML.

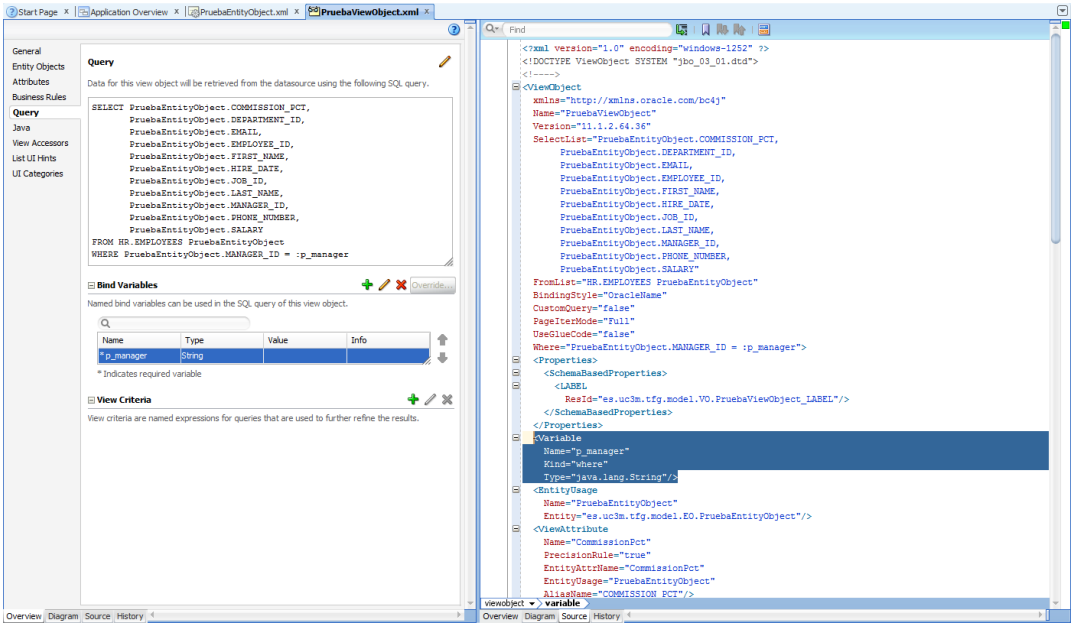

Ilustración 30. Editor de código y visual de un View Object

Las relaciones a nivel de Entity Object se realizaban a través de asociaciones, en cambio las relaciones a nivel de View Object son establecidas mediante los "*View Link*". Las diferencias entre un View Link y una asociación son prácticamente nulas, incluso al crear un nuevo View Link se puede tomar como base una asociación. La principal diferencia es que gracias a los View Links se pueden agregar al módulo de aplicación los View Objects de manera jerárquica, es decir, siguiendo la arquitectura maestro-detalle.

3.2.3.3. Application Module

Después de tener creados los View Objects específicos para la aplicación, el paso final es hacerlos accesibles para las demás capas de la aplicación y para ello deben estar contenidos en la capa de modelo. El contenedor donde deben estar los View Objects que forman parte de la capa de modelo es el módulo de aplicación.

Al ser uno de los componentes de los servicios de negocio de ADF se crea igual que todos, a través de un "*wizard*". Es posible tener varios módulos de aplicación dentro de una misma aplicación y cada uno de ellos con sus respectivos View Objects (de hecho es una práctica muy común, ya que por ejemplo se suele crear un módulo de aplicación que únicamente contiene todos los View Objects que recuperan datos para generar listas de valores), pero siempre debe haber un módulo de aplicación raíz que contenga a todos los demás.

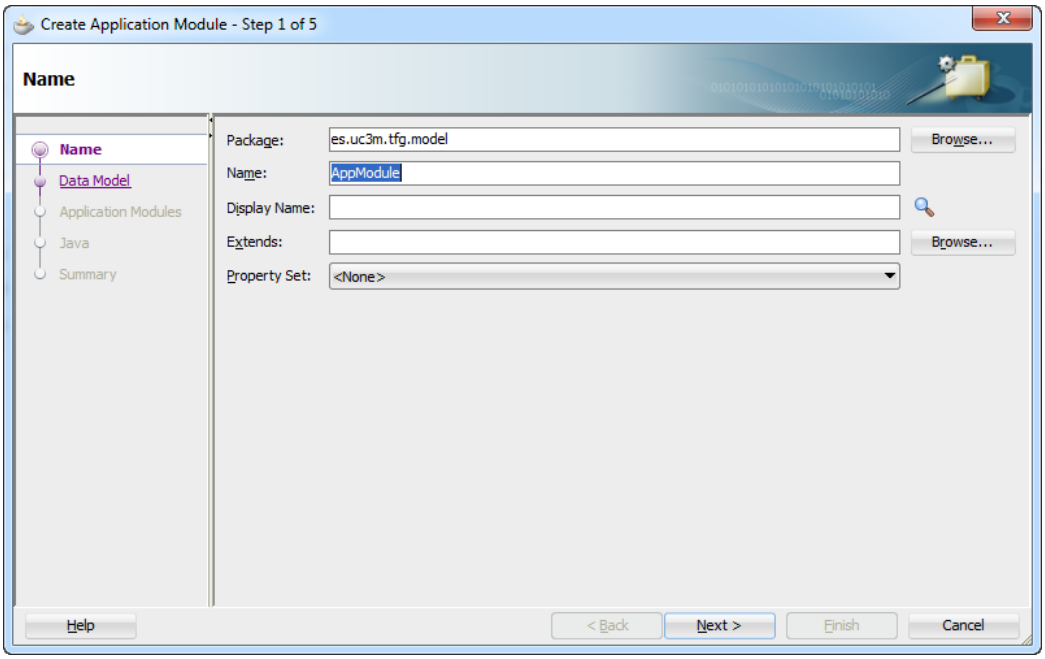

Ilustración 31. Nuevo Application Module

El siguiente paso en la creación del módulo de aplicación es generar el modelo de datos y para ello hay que agregar los View Object (y sus respectivas relaciones). Cada View Object que se agrega al módulo de aplicación genera una nueva instancia del mismo, por lo que se pueden tener varias instancias del mismo View Object dentro del mismo módulo de aplicación, incluso un mismo View Object puede generar dos instancias y que estén relacionadas como maestro-detalle.

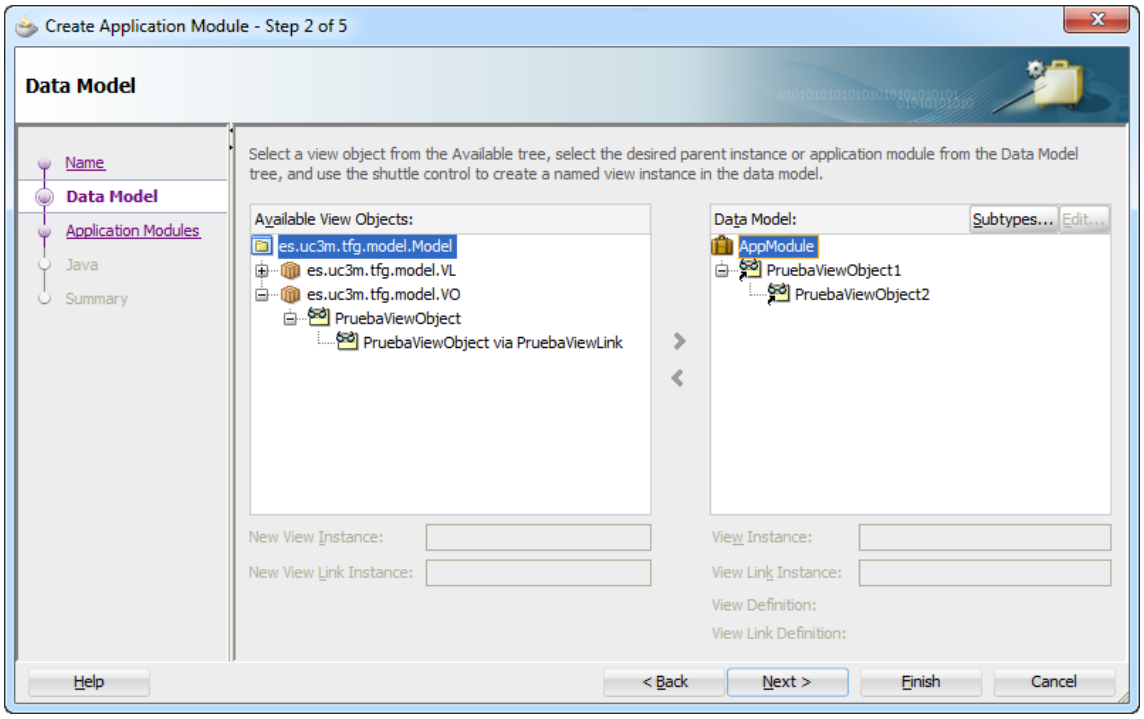

Ilustración 32. Generación del modelo de datos

En los siguientes pasos del "*wizard*" se pueden agregar otros módulos de aplicación al actual y como siempre generar los archivos Java para añadir métodos que customicen la funcionalidad por defecto. El resultado es un archivo con el nombre del módulo de aplicación que contiene el archivo XML de metadatos y los archivos Java en el caso de que hayan sido generados.

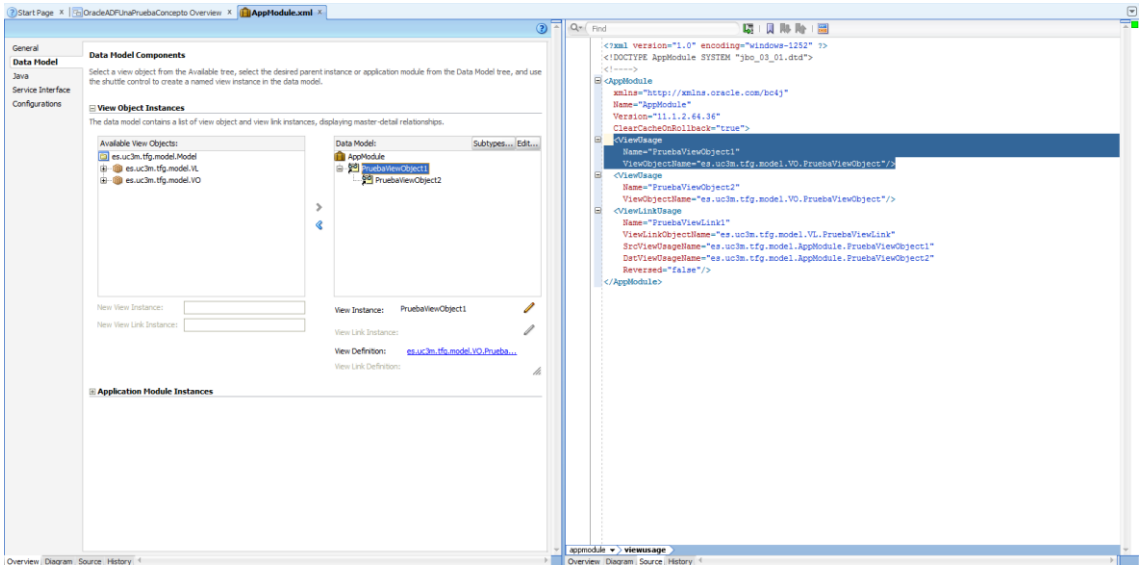

Ilustración 33. Editor de código y visual de un Application Module

En el archivo XML de metadatos se pueden configurar las características referentes al módulo de aplicación como agregar nuevas instancias de View Objects o incluso generar una interfaz de web services a través de métodos que debe crear el desarrollador en Java.

En esta breve explicación sobre los ADF Bussiness Components se ha mostrado un primer contacto de una forma más práctica para mejorar el entendimiento de las bases de estos componentes, pero en la realidad hay un gran número de operaciones adicionales que se pueden realizar. El desarrollador aprenderá a utilizar las demás características a medida que gana experiencia con el entorno de desarrollo de JDeveloper.

3.3. ADF Model

Para cualquier aplicación que siga el principio Modelo-Vista-Controlador, donde la lógica de negocio está separada de la implementación de la UI, se necesita algún modo de conexión entre ambas capas y ADF model es el responsable de este vínculo.

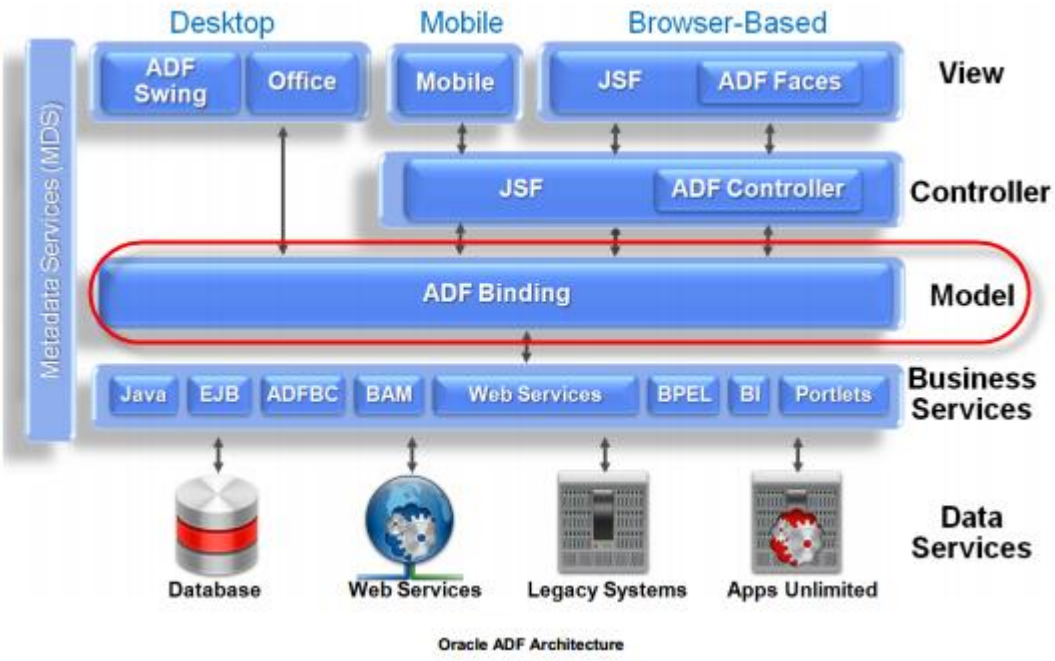

Ilustración 34. ADF Model en la arquitectura de ADF

El principal beneficio es que ADF Model proporciona una capa de abstracción

sobre la capa de servicios de negocio, por lo que independientemente de cómo son implementados (ADF BC, web services o clases Java), las herramientas que se utilizará en la vista serán las mismas.

3.3.1. Data Controls

En JDeveloper se muestran en la paleta de Data Controls. La paleta de Data Controls se sincroniza cada vez que se realizan cambios sobre el módulo de aplicación de los ADF BC, esto incluye atributos, colecciones, métodos, operaciones sobre las colecciones e incluso métodos y operaciones del módulo de aplicación.

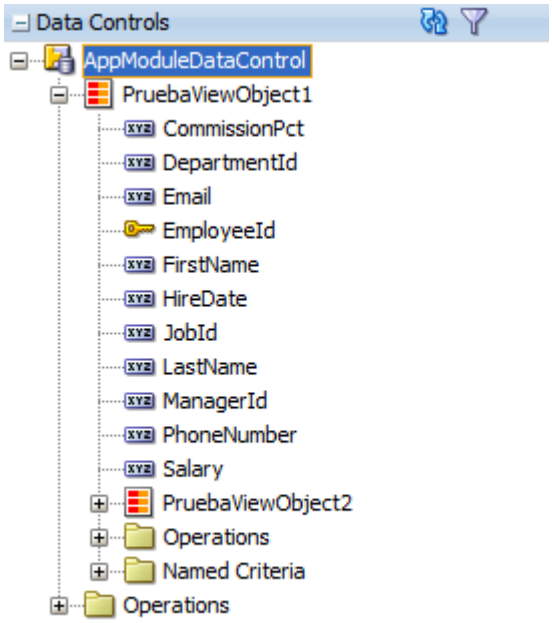

Ilustración 35. Paleta de Data Controls

Para otros tipos de servicios de negocio existe un archivo (.dcx) que se usa para almacenar la descripción del servicio. Cuando un servicio de este tipo es creado o actualizado se debe utilizar la herramienta "*Create Data Control*" para modificar el archivo .dcx.

ADF Data Controls consiste en una definición XML que describe el control, en el caso de ADF BC es el archivo XML del módulo de aplicación. Para los demás, la definición XML se crea cuando se usa la herramienta "*Create Data Control*"

Las colecciones, atributos, operaciones y named criteria de los Data Controls se pueden arrastrar directamente sobre la interfaz del usuario en forma de componentes que muestran el elemento que se desea. Por ejemplo, crear un formulario de búsqueda sobre un View Object es tan sencillo como arrastrar la instancia del View

Object que se desea sobre la UI, seleccionando formulario de búsqueda.

3.3.2. Data Bindings

Los Data Bindings son el vínculo entre los Data Controls y la UI, por ello cada página de la interfaz del usuario tiene un archivo de definición de los ADF Bindings propio. Los ADF Bindings pueden ser creados de diferentes maneras, la más sencilla y directa es el ejemplo que se ha descrito en los Data Controls. Cuando se arrastra algún elemento de la paleta de Data Controls sobre la UI, el JDeveloper genera automáticamente los ADF Bindings necesarios. También pueden ser creados de manera manual por parte del desarrollador.

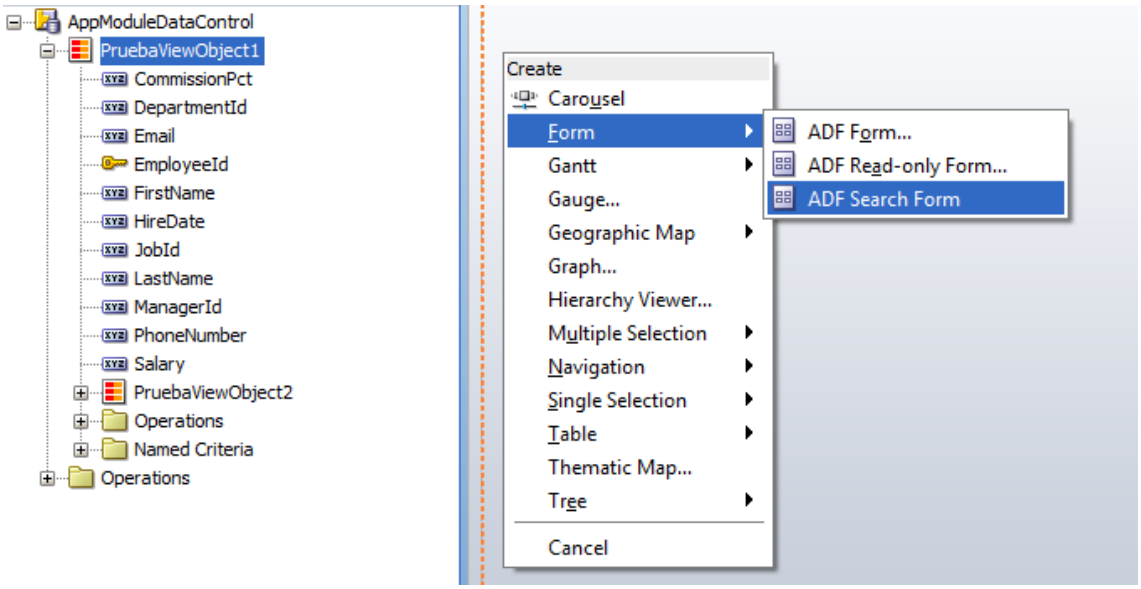

Ilustración 36. Creación ADF Bindings mediante drag and drop

El resultado de la acción anterior es un formulario de búsqueda con los campos de PruebaViewObject1, es importante añadir que aunque PruebaViewObject1 y PruebaViewObject2 se han instancias del mismo View Object son totalmente independientes por lo que si se busca una fila con el formulario de búsqueda recién creado, la búsqueda se realizara únicamente en PruebaViewObject1.

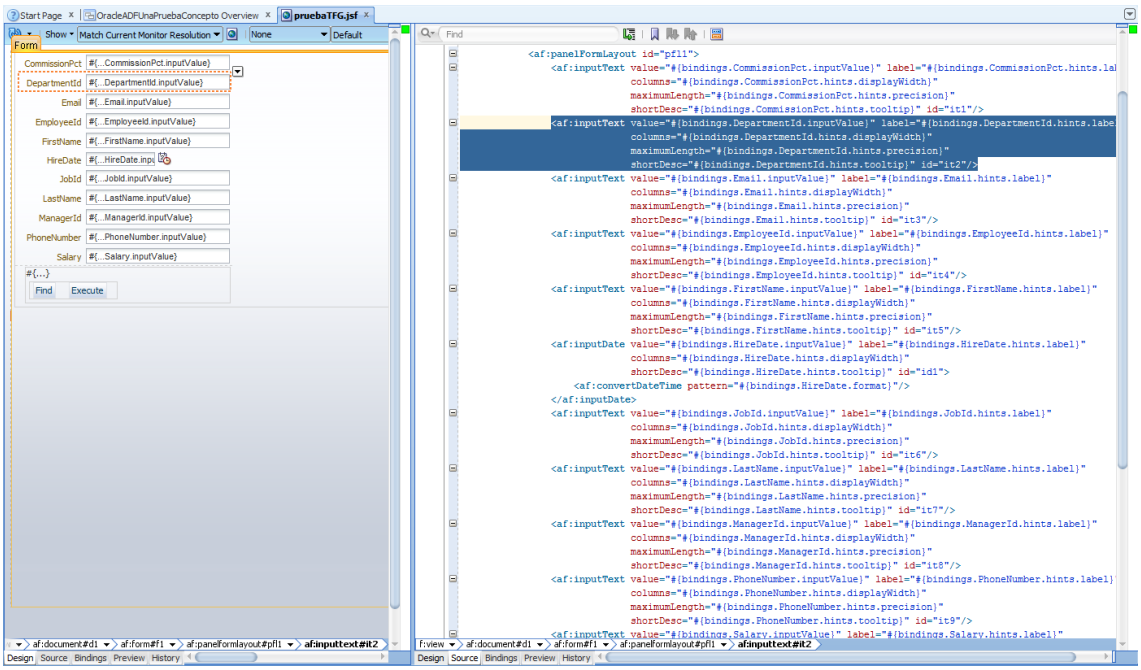

Ilustración 37. Formulario de búsqueda

En la ilustración superior se puede observar la interfaz de usuario del formulario de búsqueda creado y su código. En el código se puede ver las referencias a los bindings generados por las propiedades de cada uno de los componentes del formulario.

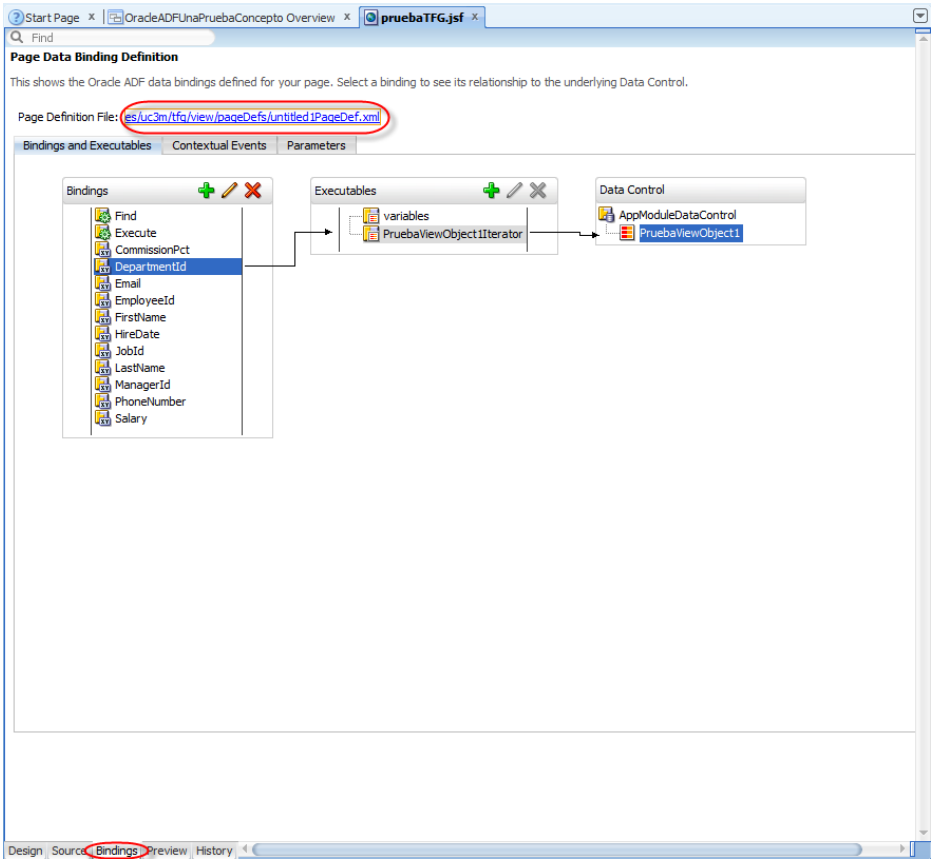

Ilustración 38. ADF Bindings generados tras realizar drag and drop

Al tener cada página de la interfaz de usuario su propia página de definición de los ADF Bindings, es posible visualizarlo y modificarlos de manera declarativa en la pestaña "bindings". Si analizamos la ilustración anterior vemos 3 contenedores dentro de la pestaña de bindings:

- **Data Control:** son los Data Controls sobre los que se basan los bindings, y son mostrados de manera jerárquica en función de las relaciones que hay entre los distintos ADF BC usados.
- **Executables:** son los intermediarios entre los bindings y los Data Controls, los más utilizados son los iteradores.
- **Bindings:** son todas las operaciones y atributos con los que la vista necesita interactuar, en la ilustración sólo se ven operaciones (find, execute) y atributos. Hay otros tipos de bindings que no se han usado en este ejemplo, tales como de tipo árbol, de tipo lista, de tipo gráfico, de tipo tabla, y otros.

En la parte superior se puede ver la ruta del archivo de definición de los ADF Bindings donde se puede modificar tanto de manera declarativa como por código.

3.4. ADF Controller

ADF Controller es el encargado de la navegación en las aplicaciones web de Oracle Fusion y ofrece un enfoque modular para diseñar el flujo de aplicación. ADF Controller está construido sobre el controlador de flujo de JSF por lo que los desarrolladores pueden utilizar las características de éste, pero ADF va más allá y con el fin de mejorar la reusabilidad, el término "page flow" utilizado en JSF se cambia por "task flow", que además de proporcionar navegación entre paginas también puede incluir navegación independiente de la UI, llamadas a métodos, enrutamientos, y llamadas a otros subflujos.

3.4.1. Task Flows

El concepto de ADF Task Flows fue introducido en la versión 11g hasta entonces los task flows eran más parecidos al flujo entre páginas de JSF. Como ya se ha explicado los task flows se dividen en dos tipos, unbounded y bounded. A continuación se muestra una breve explicación práctica de ambos.

3.4.1.1. Unbounded Task Flows

Los Unbounded Task Flows son usados generalmente como el punto de entrada a la aplicación ya que no pueden ser llamados desde otros Task Flows.

[18] ADF Training | How to Create Navigation Panes using a Menu Model in Oracle ADF. Disponible: http://www.youtube.com/watch?v=cInX_FMZJaQ

En el ejemplo se muestra como crear un menú de navegación haciendo uso de un Unbounded Task Flow, y a partir de él crea una plantilla con la que se pueden crear las demás páginas .jsf

3.4.1.2. Bounded Task Flow

Son los más utilizados, y los que forman la verdadera navegación de la aplicación. En las aplicaciones web Fusion, un Bounded Task Flow puede ser llamado desde una URL, desde otro Task Flow o incluso referenciado desde un ADF Task Flow Binding usado para las regiones.

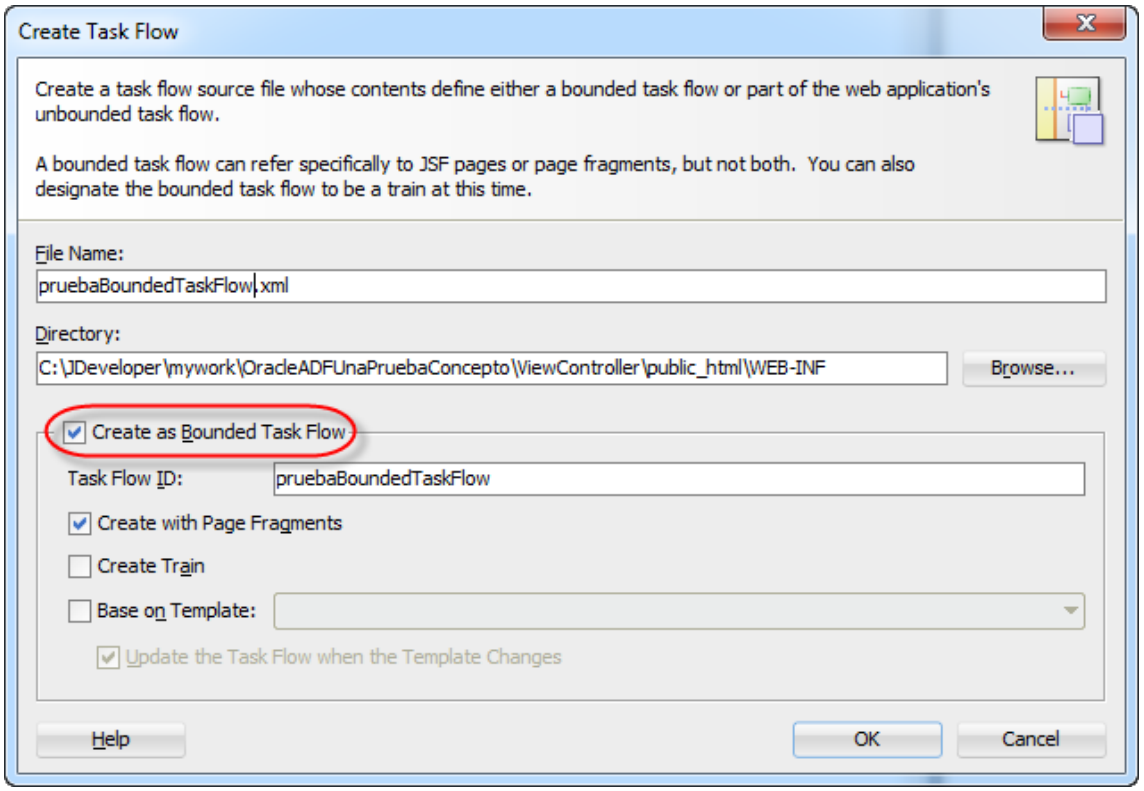

Ilustración 39. Crear nuevo Bounded Task Flow

El resultado es un nuevo archivo XML con el nombre que le hemos dado en la carpeta WEB-INF del proyecto ViewController. En él a parte de contener la navegación entre páginas de la aplicación se pueden realizar llamadas a métodos, enrutamiento o llamadas a otros Task Flows.

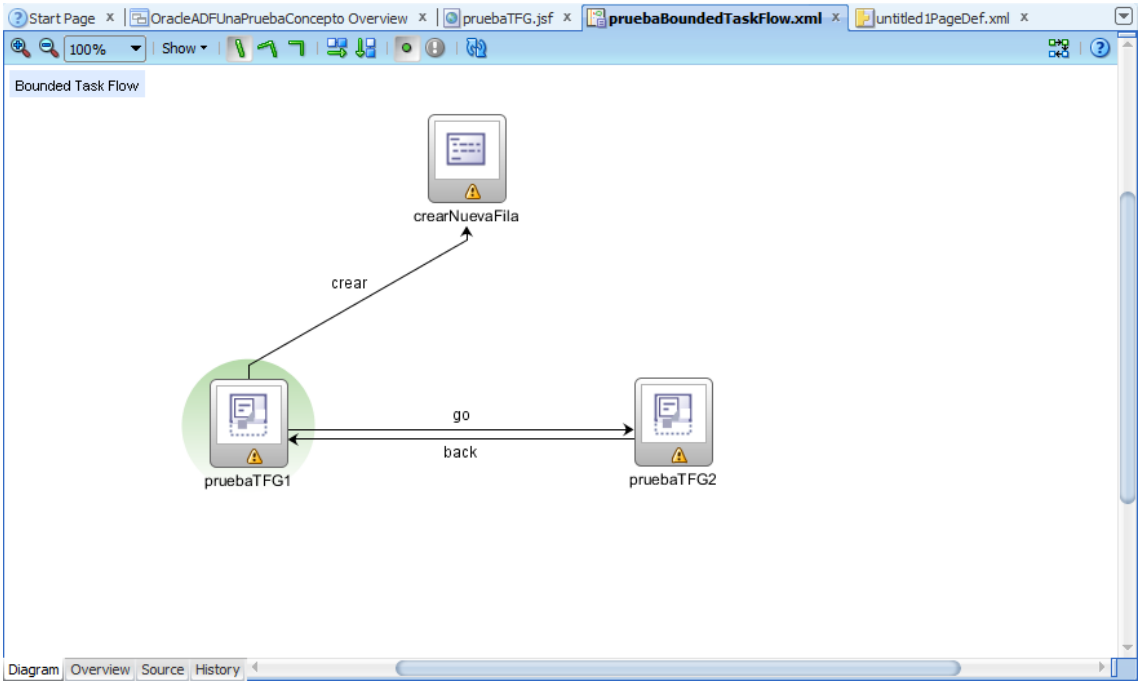

Ilustración 40. Ejemplo Bounded Task Flow

Típicamente cada uno de los flujos entre páginas estará asociado a una acción del usuario, tal como hacer click en un botón. En el Task Flow creado observamos 3 elementos:

- **pruebaTFG1**: es una página que permite dos acciones, con la acción go se navegará a la página TFG2, y con la acción crear se llamara al método crearNuevaFila
- **pruebaTFG2**: es una página que viene de pruebaTFG2 y puede volver atrás con la acción back.
- **crearNuevaFila**: es un método llamado desde pruebaTFG1 que creará una nueva fila en un View Object.

3.4.2. ADF Security

En el desarrollo de aplicaciones Fusion, Oracle ADF Security proporciona distintas características preconstruidas que facilitan a los desarolladores crear aplicaciones ADF seguras. La seguridad se realiza en las páginas y task flows, automáticamente se realiza una comprobación de seguridad JAAS cada vez que un usuario intenta acceder a un bounded task flow, por lo que no se requiere ningún tipo de codificación para asegurar estos elementos. Además provee un servlet de autenticación que delega la autenticación al contenedor de seguridad de J2EE, pero permite al desarrollador de la aplicación el control exacto de cuando el usuario realiza

la acción de login/logout.

Esta es la secuencia que se realiza para las aplicaciones protegidas con ADF Security:

1. Un usuario realiza una petición un página (que internamente llamará a un bounded task flow)
2. La capa de ADF Security comprueba si la seguridad está habilitada en la configuración de la aplicación ADF
3. Si la seguridad está habilitada, la capa de seguridad comprueba si la seguridad está habilitada para autenticación solo o para autorización también.
4. Si la autorización es obligatoria, ADF Security comprueba si existe un usuario anónimo principal y si los permisos concedidos a él son suficientes para ejecutar la página (páginas públicas).
5. Si el acceso a la página no es posible con los privilegios del usuario anónimo, el framework lanza la autenticación a través de redireccionar la petición al servlet de autenticación de ADF.
6. El servlet delega la petición de autenticación al contenedor de J2EE
7. Usando el servidor de Oracle Weblogic, el contenedor responde a la petición con un formulario o enviando la cabecera de la respuesta que permite al navegador mostrar su formulario de login
8. Los credenciales proporcionados por el usuario se comprueban contra el proveedor de autenticación del servidor Weblogic. Si la autenticación es satisfactoria, el servidor redirecciona la petición al servlet de autenticación. La sesión está ahora autenticada y se recogen los datos del usuario y del grupo empresarial al que pertenece.
9. Si la seguridad ADF está configurada para usar un único punto de entrada para los usuarios autenticados, el servlet de autenticación envía la petición a esta página. Si no, directamente se envía al usuario a la página a la que se ha realizado la petición original.

ADF Security proporciona expresiones Java y Expression Language (EL) que permiten a los desarrolladores a conseguir de manera muy sencilla información sobre el usuario actual, incluyendo sus roles y permisos. Aunque se pueden dar de alta usuarios y roles propios de la aplicación, lo más habitual es utilizar Lightweight Directory Access Protocol (LDAP). Por ello Oracle Weblogic Server permite la autenticación a través de distintos proveedores.

El primer paso para asegurar una aplicación con ADF Security es configurar el servidor Weblogic para que actúe como proveedor de la autenticación de los usuarios, este paso a va ser omitido y se centrará en la configuración de ADF Security.

Ilustración 41. Configuración ADF Security

JDeveloper permite la configuración de ADF Security en primer lugar a través de un wizard, en el que se elegirán las características básicas. El primer paso del wizard para configurar la seguridad de ADF es elegir las características que se quieren habilitar, autenticación y autorización o solo autenticación.

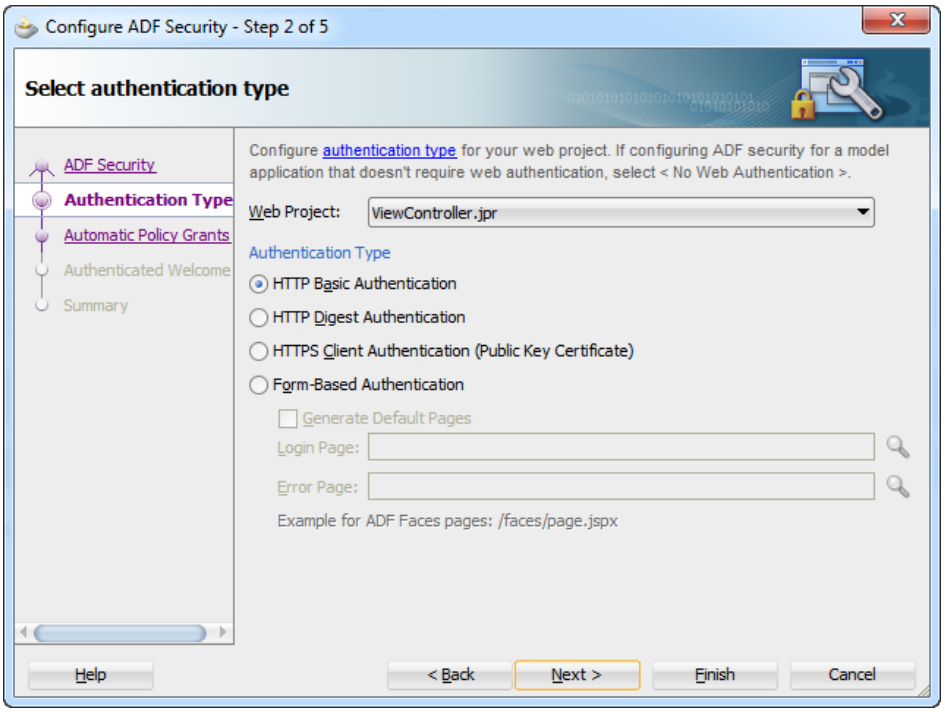

Ilustración 42. Tipo de autenticación

En el siguiente paso se elige el tipo de autenticación que se quiere utilizar hay 4 posibles opciones:

- **HTTP basic Authentication:** el navegador lanza un diálogo propio para la autenticación del usuario
- **HTTP Digest Authentication:** es similar anterior excepto que la contraseña es encriptada.
- **HTTPS Client Authentication:** la autenticación se realiza por un certificado configurado en el navegador del cliente
- **Form-Based Authetication:** la autenticación se realiza usando un formulario HTML embebido en archivo HTML, JSP o JSF que utiliza la acción "*j_security_check*".

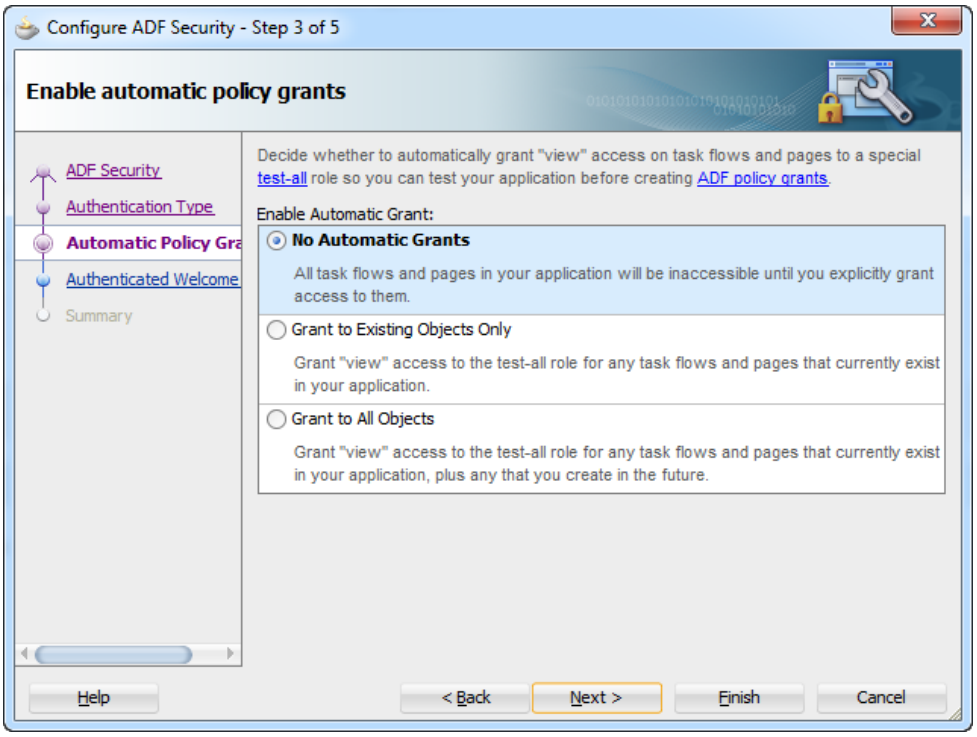

Ilustración 43. Selección de la política de permisos

En este paso se selección la política de permisos que se configurará. La primera opción "*No Automatic Grants*" configura la aplicación para no permitir acceso a ningún objeto hasta que no sea especificado manualmente. La segunda opción "*Grant to Existing Objects Only*" concede el permiso de visuzalición al rol de aplicación creado por defecto "*test-all*" a todos los objetos existentes. La última opción "*Grant to All Objects*" es similar a la anterior pero los permisos los concede a los objetos existentes y a los creados posteriormente.

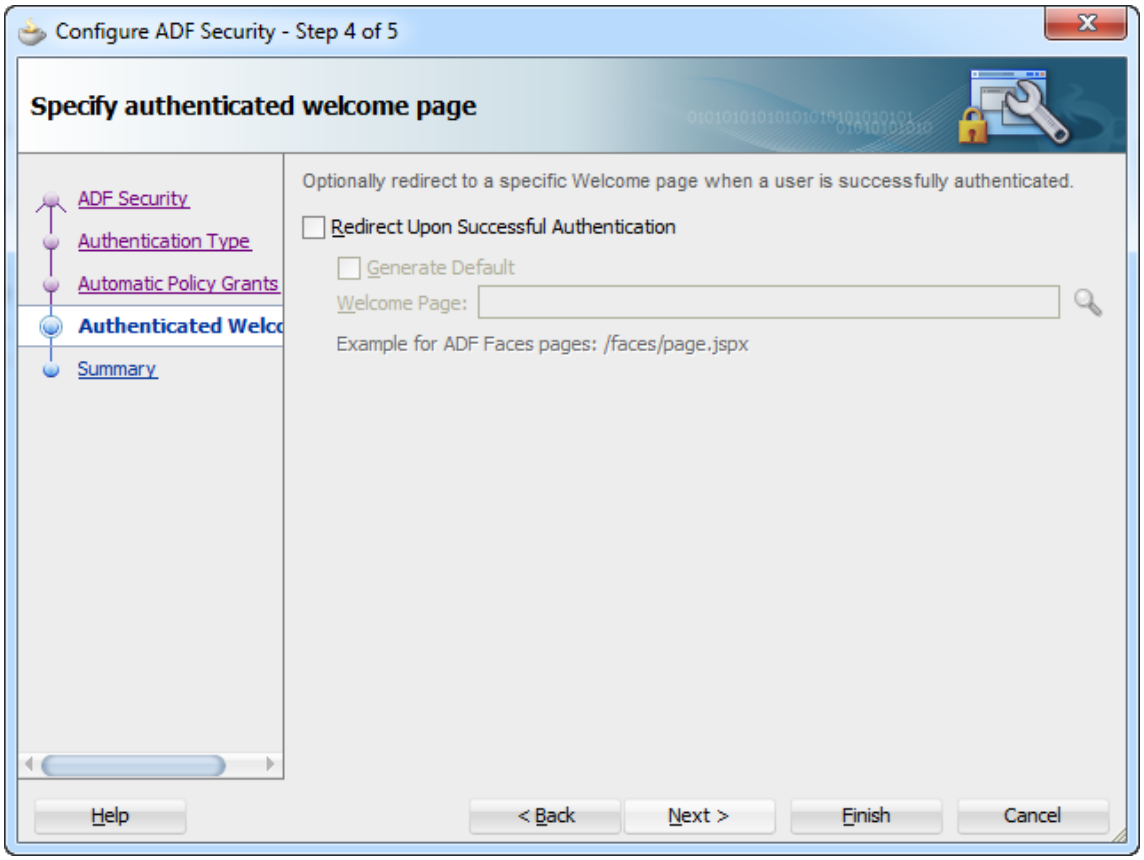

Ilustración 44. Especificar página de bienvenida

En este último paso se configura o genera una página como página de inicio para todos los usuarios autenticados en la aplicación. Da igual la página a la que quiera acceder el usuario, que en después de la autenticación el usuario será redireccionado a la página definida en este paso.

Tras completar el "*wizard*", se realizan un conjunto de cambios en los siguientes archivos.

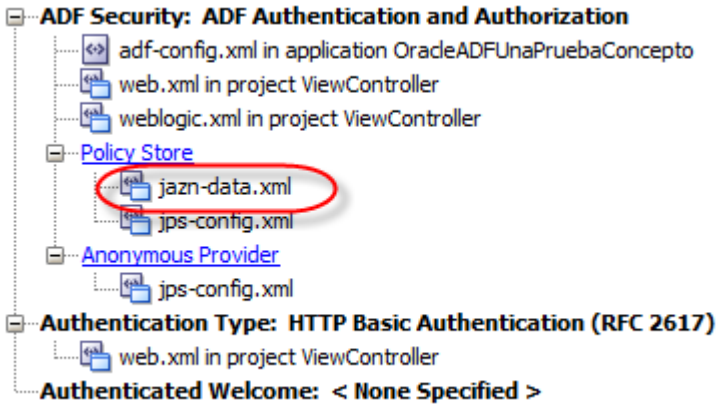

Ilustración 45. Archivos cambiados tras la configuración de ADF Security

El archivo jazn-data.xml es el más importante en la configuración de ADF Security, en él se almacenan la política de permisos seleccionada y los permisos que tiene cada rol.

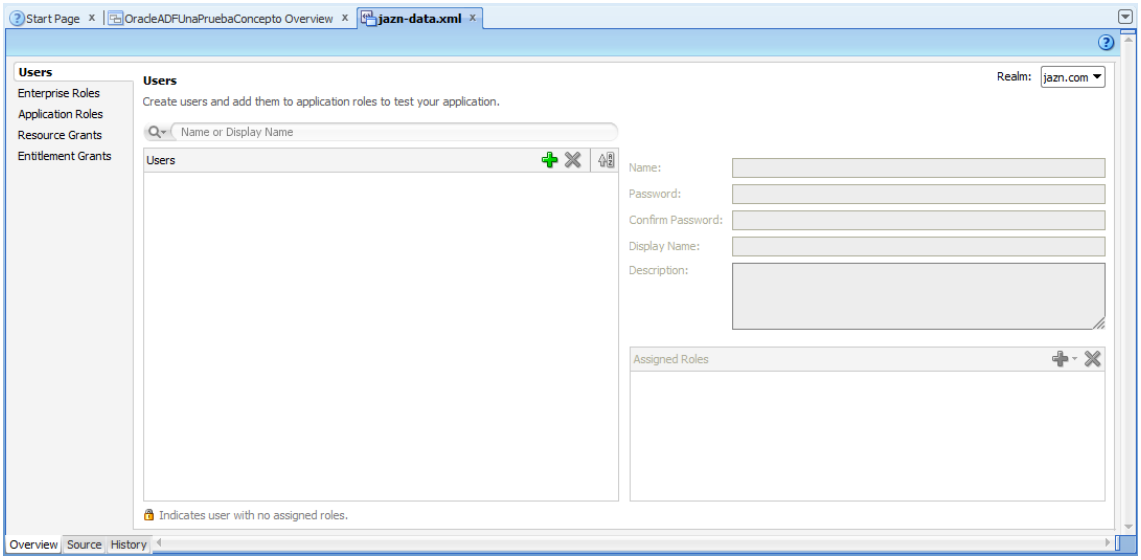

Ilustración 46. Archivo jazn-data.xml

Dentro de la seguridad de ADF hay dos tipos de roles, Enterprise Roles y Application Roles. Los Enterprise Roles mapean grupos de usuarios dentro de una organización, es decir, los grupos recuperados por el servidor Weblogic. Los Application Roles son roles a nivel de aplicación y son mapeados con los Enterprise Roles, permitiendo relaciones de 1 a 1.

Los Application Roles tiene un nivel más fino de granularidad, ya que sólo son existentes en una única aplicación, en cambio los Enterprise Roles son roles a nivel de organización, por lo que pueden ser utilizados en diferentes aplicaciones.

Supongamos que la aplicación tiene el requisito de seguridad de crear un rol llamado ADMIN y se debe asegurar cierta página. En lenguaje de JDeveloper este tipo de rol se refiere a un Application Role. Si sólo se crea este rol, cuando un usuario accediera al sistema, Weblogic recuperaría su grupo empresarial (digamos ADMINISTRATOR), pero el servidor de aplicación no tiene forma de saber cuál es la relación entre ADMIN y ADMINISTRATOR. Para ello se mapea en la aplicación el Application Role ADMIN, con el Enterprise Role ADMINISTRATOR.

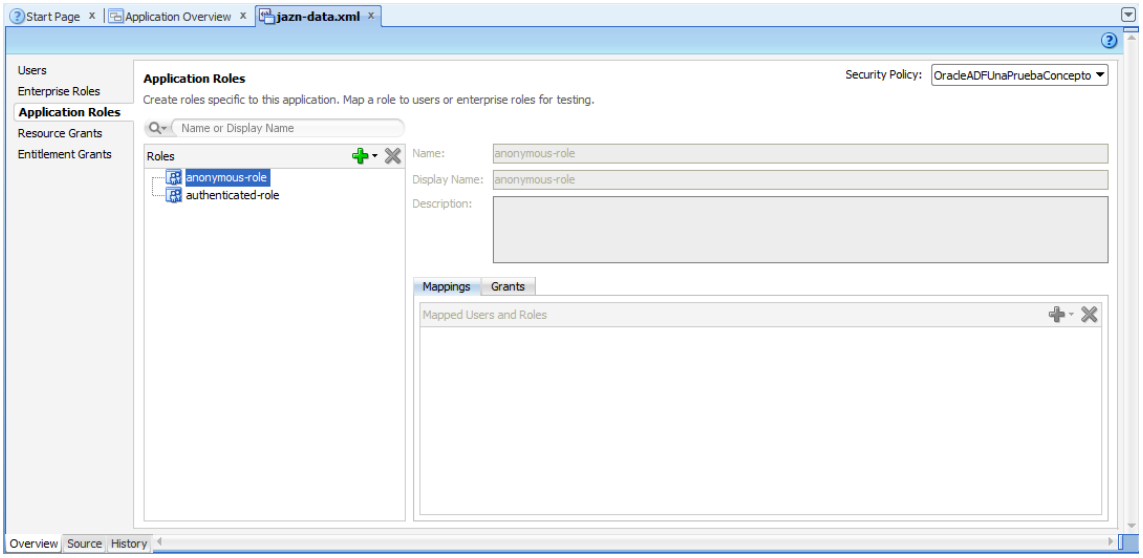

Ilustración 47. Application Roles

Por defecto ADF Security tiene creados dos Application Roles, "*anonymous-role*" y "*autheticated-role*". Así que el primer paso para asegurar la aplicación sería la creación y la configuración de los distintos tipos de roles que necesitará la aplicación.

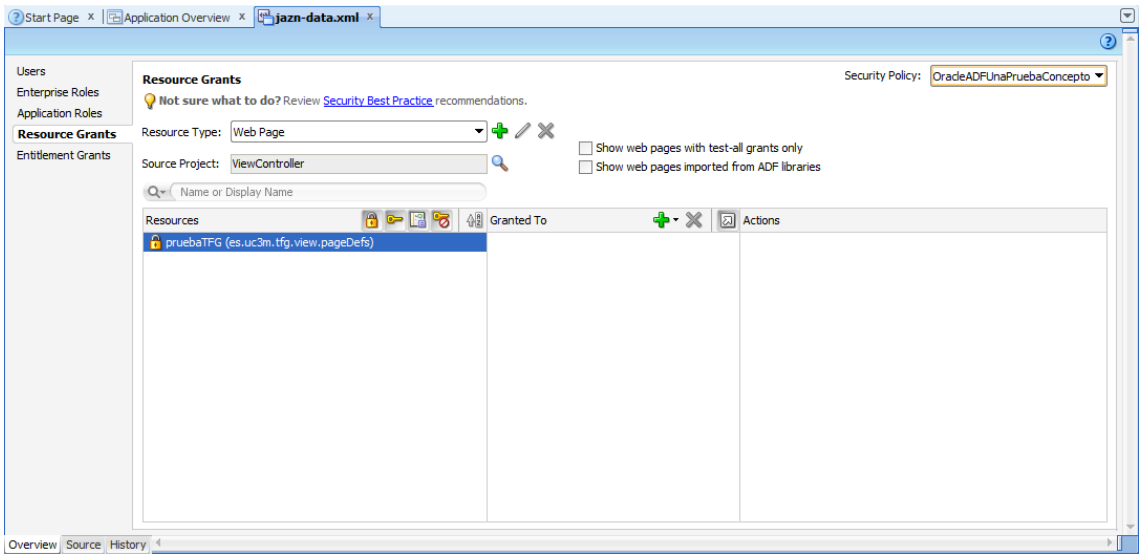

Ilustración 48. Gestión de permisos

Una vez los roles han sido correctamente configurados, el siguiente paso es proporcionar los permisos a los distintos roles para ver o modificar los diferentes objetos (task flows, Entity Object, páginas). Es importante resaltar que sólo las páginas que hacen uso de los ADF Bindings pueden ser aseguradas.

3.5. ADF Faces

Oracle ADF Faces rich client (más conocido como ADF Faces) es un conjunto de componentes JSF que incluyen funcionalidades AJAX preconstruidas. Mientras que AJX proporciona una sensación de funcionalidad en el lado del cliente a aplicaciones basadas en navegador, JSF proporciona control en el lado del servidor, lo que reduce la cantidad de código JavaScript necesario para implementar aplicaciones basadas en AJAX.

JDeveloper junto con Oracle ADF Faces proporciona una gran cantidad de tareas que se pueden desarrollar de manera declarativa, generando código que incluye expresiones EL y la generación de los bindings.

ADF Faces ofrece la funcionalidad necesaria en "*Rich Internet Applications (RIA)*" y además una librería de más de 150 componentes.

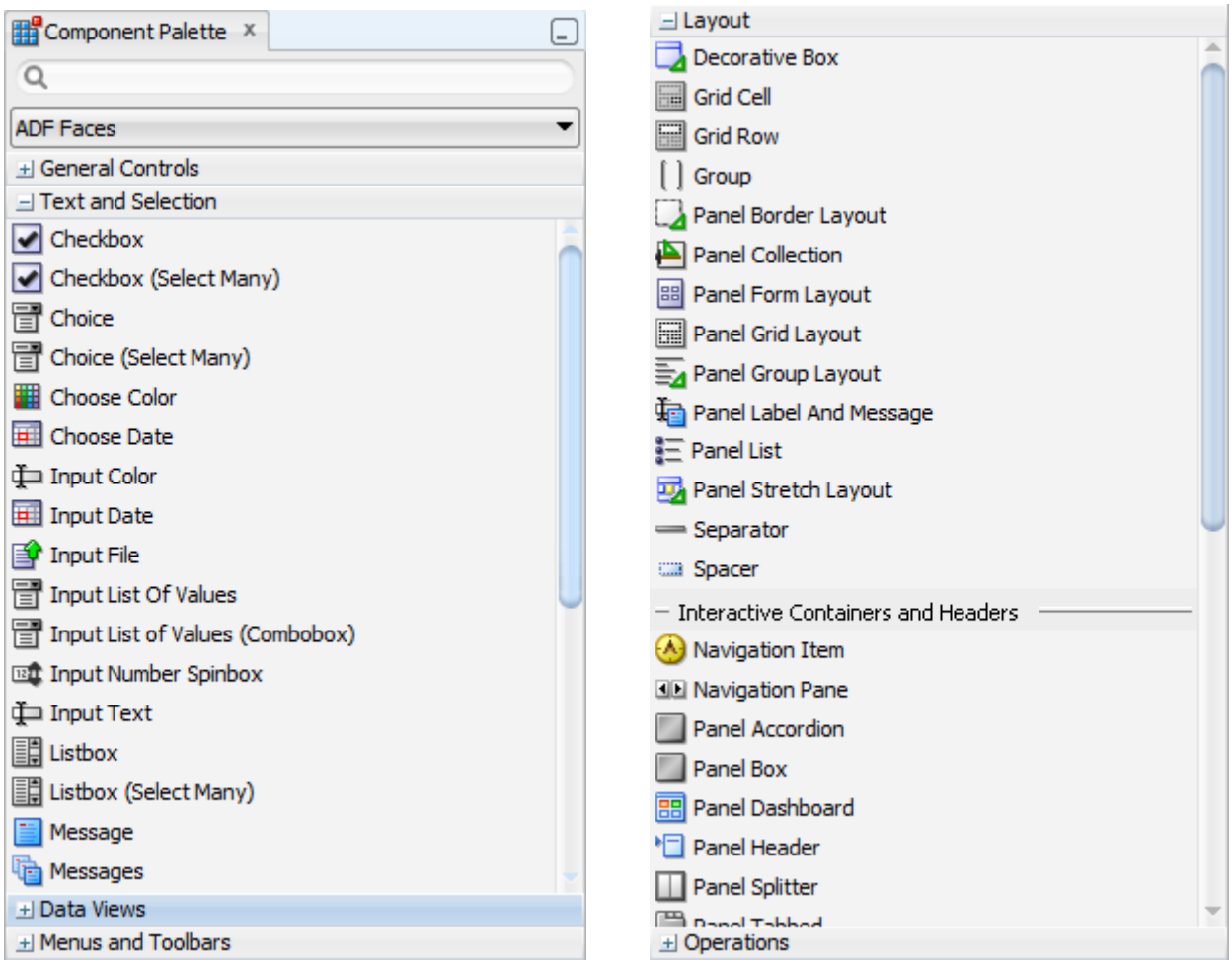

Ilustración 49. Paleta de componentes de ADF Faces

En la ilustración superior se puede observar un pequeño conjunto de los componentes que ofrece ADF Faces. Anteriormente se explicó que gran parte de los componentes pueden ser agregados arrastrando servicios de negocio desde la paleta de Data Controls a la página, pero además se pueden agregar arrastrando desde la paleta de componentes.

Cada componente tiene unas propiedades que pueden ser cambiadas sin necesidad de acceder al código, esto se puede realizar gracias al inspector de propiedades.

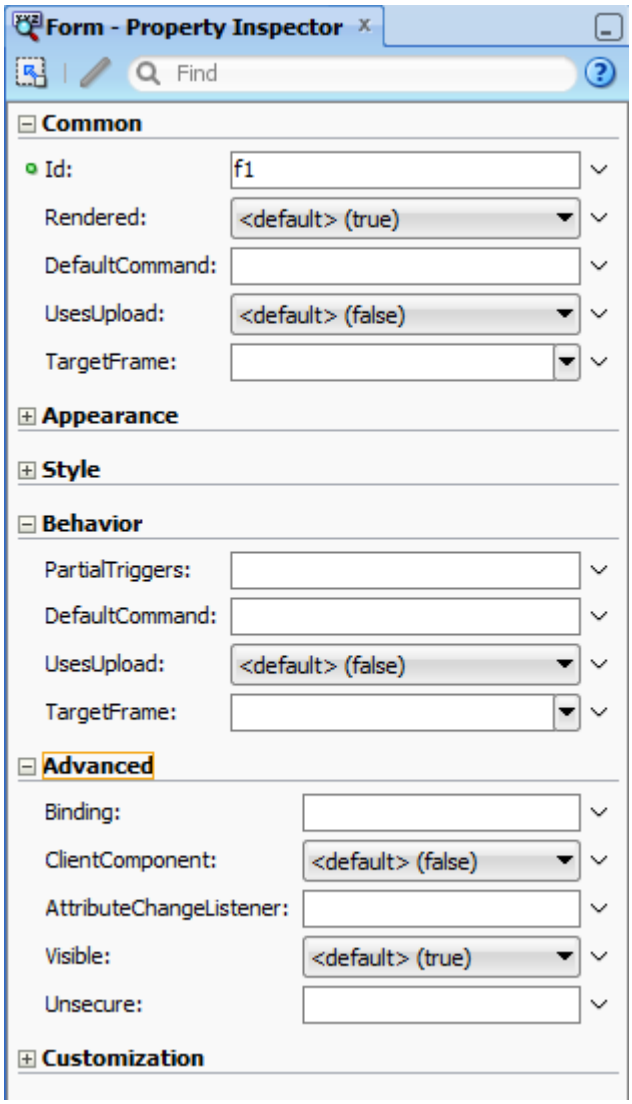

Ilustración 50. Inspector de propiedades

El inspector de propiedades muestra todos los tipos de configuración del componente, tanto visuales como funcionales, incluso se pueden agregar operaciones de los ADF Bindings o métodos de un Backing Bean como manejadores de eventos del componente.

4. Prueba de Concepto

4.1. Introducción

Tras el análisis del framework ADF, la mejor forma de afianzar los conocimientos obtenidos y de observar las capacidades del mismo, es a través de la práctica. Para ello se inició el desarrollo de una aplicación web para una empresa que encaja perfectamente en el perfil para las que ADF ha sido diseñado.

La empresa para la que se va a realizar la aplicación es una consultoría española (Arin Innovation S.L.) con sede principal en Bilbao y distintas subsedes en Madrid, Londres y Colombia destinada a presentar soluciones de negocio para pequeñas y medianas empresas, sobre todo a tecnologías de Oracle (Oracle E-Bussiness Suite, Oracle JD Edwards Enterprise). Como consultoría tiene una gran cantidad de clientes a los que ofrece sus servicios, y a los que tiene que facturar sus servicios en función de las horas que emplean sus trabajadores en las distintas empresas.

Para mantener este control existe una aplicación web que principalmente permite a los empleados introducir las horas que emplea a los clientes cada jornada laboral y obtener informes de las horas empleadas en cada cliente a los trabajadores encargados de realizar la facturación de las horas. Aunque el objetivo principal es el comentado anteriormente, se necesita una parte de administración con la que se configure la aplicación web (agregar nuevos clientes a los que realizar una imputación de horas, agregar nuevos proyectos a los clientes ya existentes y otras muchas opciones que son necesarias).

Esta aplicación actualmente se denomina AriNet y está desarrollada con Java Server Pages, pero debido a que la empresa trabaja con tecnologías de Oracle, se desea crear una aplicación similar que cubra los mismos requisitos de usuario pero haciendo uso del framework ADF.

El proyecto que se llamará AriNet 2 se va dividir en tres fases debido al gran tamaño de la aplicación web, la primera fase es la recogida en este documento, e incluye la mayor parte de los requisitos usuario de la aplicación.

4.2. Diseño del proyecto con ADF

Para realizar el diseño del proyecto con el framework ADF se va a ir de abajo a arriba, es decir, se empezará con los servicios de negocio y se terminará con la interfaz de usuario. Durante este apartado se tendrán en cuenta los requisitos de la aplicación, ya que condicionarán la toma de decisiones en cada una de las capas en las que ADF separa las aplicaciones.

4.2.1. Servicios de Negocio

Los datos que utilizará la aplicación (y que utiliza actualmente la otra aplicación) se encuentran una base de datos SQL de Oracle con el siguiente diagrama entidad-relación.

La capa de servicios de negocio maneja la interacción con la fuente de los datos que la aplicación maneja y proporciona la persistencia necesaria para no corromper los mismos. La tecnología más recomendable cuando se utiliza ADF (ya que ha sido desarrollada íntegramente para este framework) es ADF Bussiness Componentes y por ello ha sido la seleccionada para desarrollar está capa.

Como la estructura de tablas, y el modelo relacional de la base de datos ya está desarrollado, se debe diseñar la correcta interacción entre la base de datos y la aplicación haciendo uso de ADF BC.

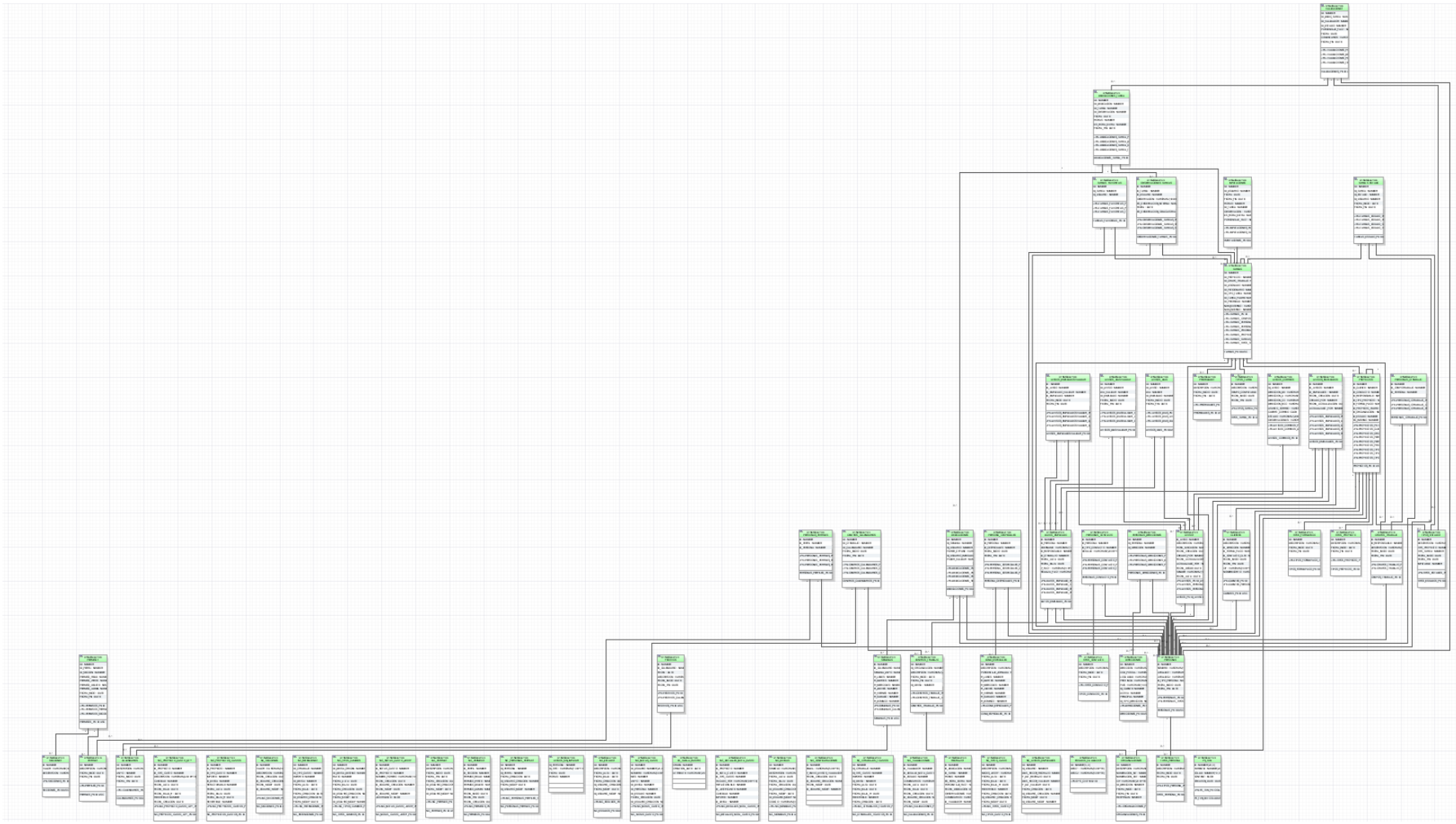

Ilustración 51. Diagrama E-R de la base de datos

Primero hay que distinguir la forma de interacción con las tablas de la base de datos que van a ser utilizadas en la aplicación, ya que los componentes que van a ser necesarios dependen de si serán modificadas o si serán de sólo lectura. Para insertar o modificar registros de una tabla se necesita crear un Entity Object que se mapee con la tabla de la base de datos. Los Entity Objects solamente pueden ser mapeados con una única tabla, por lo que por cada tabla en la que se quiera insertar o modificar registros deberá tener su Entity Object asociado, no es necesario mapear todos los campos de la tabla con el Entity Object solamente los que se quieren modificar.

La capa de modelo no puede trabajar con Entity Objects sólo con View Objects, por ello es necesario crear View Object basados en los Entity Objects ya creados. En este caso no es necesario crear un View Object por cada Entity Object, sino que se puede crear un View Object basado en varios, siempre y cuando tengan una relación entre ellos.

Una vez se tienen los Entity Objects y View Objects que mapean los datos que pueden ser modificados, viene el turno de los de sólo lectura. Éstos son más difíciles de planificar ya que pueden ser los resultados de un formulario de búsqueda o incluso una lista de valores que se cargue en un componente desplegable, pero también tienen el beneficio de poder agregarse en un futuro con mayor flexibilidad sin cambiar la lógica de negocio.

En este punto se tendrían todos los objetos que interactuarían con la base de datos pero de una forma independiente, es decir, sin ninguna relación, cuando en una base de datos las relaciones entre las distintas tablas es muy común. Hay dos formas de relacionar los ADF BC entre ellos, si es entre Entity Objects la relación se realiza mediante Association y si es entre View Objects mediante un View Link, aunque la diferencia práctica entre ambas es ínfima. Relacionando los distintos componentes de los servicios de negocio se consiguen comportamientos de maestro-esclavo que es muy útil y facilita mucho el desarrollo.

Por último como nexo de unión entre los servicios de negocio y la capa de modelo se crea el módulo de aplicación, dentro de éste pueden ser creados otros módulos de aplicación hijos.

A continuación se muestran tablas de los distintos ADF Bussiness Components utilizados en la fase 1 de la aplicación AriNet 2, mostrando el nombre del objeto, el origen de los datos y el requisito por el que ha sido creado.

Entity Objects

Nombre	Origen	Requisito
CentrosCalendarioEO	CENTROS_CALENDARIO	RF-14
CentrosTrabajoEO	CENTROS_TRABAJO	RF-14
ClientesEO	CLIENTES	RF-14
CondEspecialesEO	COND_ESPECIALES	RF-14
DatosEmpleadoEO	DATOS_EMPLEADO	RF-14
DireccionesEO	DIRECCIONES	RF-14
GruposTrabajoEO	GRUPOS_TRABAJO	RF-14
ImputacionesEO	IMPUTACIONES	RF-07
OrganizacionesEO	ORGANIZACIONES	RF-14
PerfilesEO	PERFILES	RF-14
PersonaCespecialesEO	PERSONA_CESPECIALES	RF-14
PersonaPerfilEO	PERSONA_PERFIL	RF-14
PersonasEO	PERSONAS	RF-14
PersonasGTrabajoEO	PERSONAS_GTRABAJO	RF-14
ProyectosEO	PROYECTOS	RF-14
TareasEO	TAREAS	RF-14
TiposProyectoEO	TIPOS_PROYECTO	RF-14
TiposTareaEO	TIPOS_TAREA	RF-14

View Objects

Nombre	Origen	Requisito
CentrosCalendarioVO	CentrosCalendarioEO	RF-11
CentrosTrabajoVO	CentrosTrabajoEO	RF-11
ClientesVO	SELECT	RF-08
CondEspecialesVO	CondEspecialesEO	RF-11
ConfClientesVO	ClientesEO	RF-11
ConfProyectosVO	ProyectosEO	RF-11
ConfTareasVO	TareasEO	RF-11
DireccionesVO	DireccionesEO	RF-11
EmpleadosVO	DatosEmpleadoEO	RF-11
GruposTrabajoVO	GruposTrabajoEO	RF-11
ImputacionesVO	ImputacionesEO	RF-07
OrganizacionesVO	OrganizacionesEO	RF-11
PersonasGTrabajoVO	PersonasGTrabajoEO	RF-11
PersonasPerfilesVO	PersonaPerfilEO	RF-11
PersonasVO	PersonasEO	RF-11
ProyectosVO	SELECT	RF-08
TareasVO	SELECT	RF-08
TiposProyectoVO	TiposProyectoEO	RF-11
TiposTareaVO	TiposTareaEO	RF-11

Además de todos estos View Objects hay que sumar las listas de valores que se han creado en un nuevo módulo de aplicación, pero debido a que no proporcionan funcionalidad como tal y al gran número de ellos, no han sido mostrados en la tabla anterior.

Con crear todos los ADF Bussiness Components no es suficiente, ya que al crear cualquier componente la mayor parte de las propiedades de éste son establecidas por defecto. Por ejemplo después de crear un Entity Object la clave principal tiene como valor por defecto un valor nulo, por lo que debe ser establecido para que genere el próximo número de la secuencia que utiliza la base de datos en la clave principal de la tabla que mapea. ADF BC proporciona un sinfín opciones de configuración muy variadas, desde a nivel de atributo del Entity Object como en el ejemplo citado hasta a generar un View Accesor que son usados para definir y filtrar otros Entity Objects para usar como lista de valores.

4.2.2. Capa de Modelo

La capa de modelo es una abstracción que conecta los servicios de negocio con otras capas mediante una interfaz común a todos los servicios de negocio (independientemente de la implementación). En ella deben estar todos los datos a los que pueda acceder la capa de controlador y de flujo.

La implementación seleccionada para los servicios de negocio ha sido ADF BC por lo que para generar la interfaz con la que interaccionan otras capas de los datos necesarios se deben agregar los View Objects al módulo de aplicación raíz, ya sea como hijo directo del módulo de aplicación raíz o como hijo de otros módulos de aplicación hijos del raíz. Esto no implica que se deban agregar todos los View Objects al módulo de aplicación ya que puede ser que muchos de ellos actúen únicamente con otros View Objects como parte de la lógica de negocio y no sean utilizados en otras capas. Un ejemplo son los View Objects que actúan como listas de valores, no todos son necesarios en la capa de modelo y para los que lo son es una práctica recomendable agregarlos todos a un único módulo de aplicación distinto del raíz, favoreciendo la reutilización de los mismo en distintas zonas de la aplicación.

A continuación se muestran los View Objects agregados al módulo de aplicación para que estén disponibles en la capa de modelo de la fase 1 de la aplicación AriNet 2, mostrando el nombre del objeto, el origen de los datos y el requisito por el que ha sido creado.

Nombre	Origen	Requisito
CentrosTrabajoVO1	CentrosTrabajoVO	RF-11
ClientesLOV1	ClientesLOV	RF-11
ClientesVO1	ClientesVO	RF-08
CondEspecialesVO1	CondEspecialesVO	RF-11
ConfClientesVO1	ConfClientesVO	RF-11
ConfGruposTrabajoLOV1	ConfGruposTrabajoLOV	RF-11
ConfProyectosLOV1	ConfProyectosLOV	RF-11
ConfProyectosVO1	ConfProyectosVO	RF-11
ConfTareasVO1	ConfTareasVO	RF-11
EmpleadosVO1	EmpleadosVO	RF-11
GruposTrabajoVO1	GruposTrabajoVO	RF-11
ImputacionesVO1	ImputacionesVO	RF-07
OrganizacionesVO1	OrganizacionesVO	RF-11
PerfilesLOV1	PerfilesLOV	RF-11
PersonasVO1	PersonasVO	RF-11
TareasLOV2	TareasLOV	RF-11
TiposProyectoVO1	TiposProyectoVO	RF-11
TiposTareasVO1	TiposTareasVO	RF-11

4.2.3. Capa de Controlador y Vista

El diseño de estas dos capas se va a realizar de manera conjunta ya que están fuertemente relacionadas. El flujo de la aplicación aunque pueda trabajar a nivel de métodos también integra las distintas páginas de la aplicación en los task flows.

Para crear los task flows que definan el flujo entre las páginas de la aplicación es necesario que los requisitos de usuario estén claramente definidos, ya que en base a ello se diseñarán la vista y el controlador, y los cambios en ellos en un futuro pueden ser muy costosos.

En la aplicación AriNet 2 el flujo es muy sencillo, en los requisitos se muestra que deben existir 4 páginas (Inicio, Mis Dedicaciones, Informes, Configuración) que tienen que ser accesibles desde cualquiera, a través de un menú de navegación. Con esta pequeña descripción se puede pensar que lo primero que hay que realizar es un Unbounded Task Flow, que defina el menú de navegación. El menú puede ser creado en el task flow por defecto adfc-config.xml o crear un archivo nuevo pero el resultado es el mismo, en este caso se ha utilizado el archivo adfc-config.xml para crear el menú.

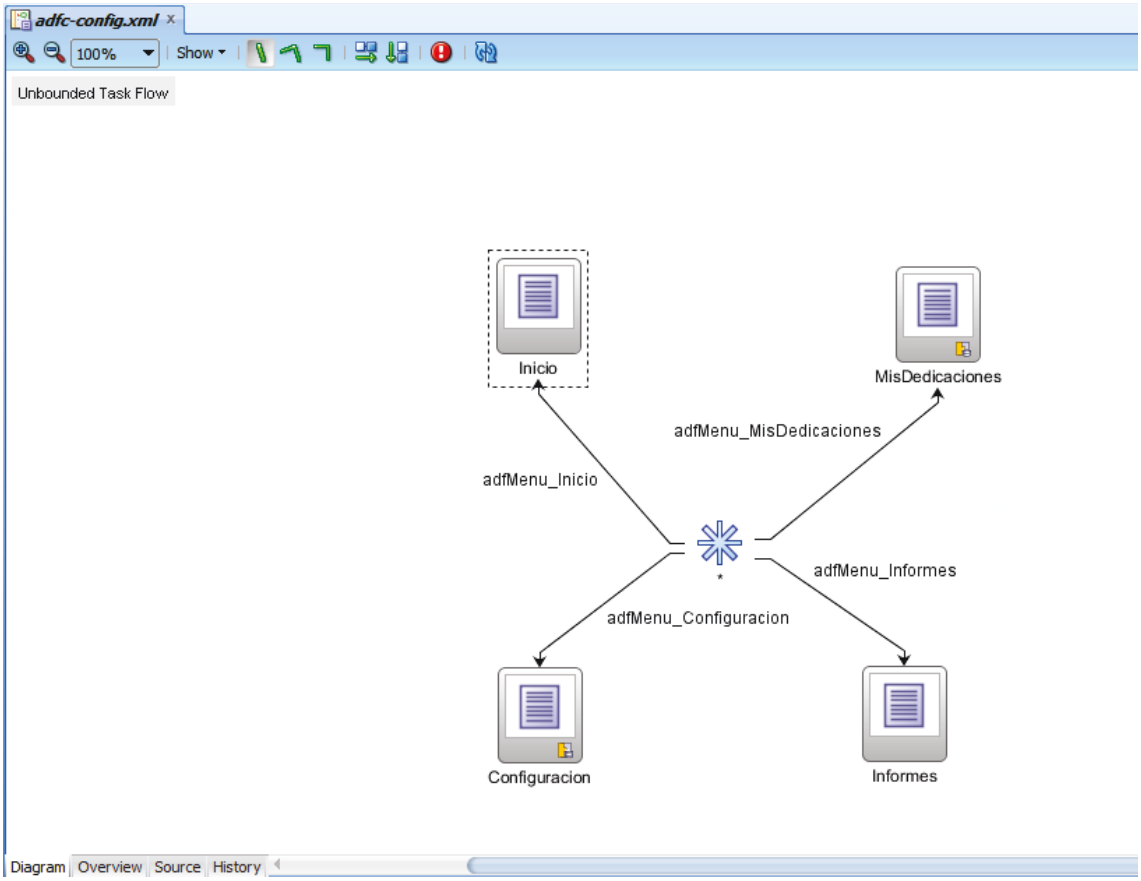

Ilustración 52. Unbounded task flow para definir el menu de navegación

A continuación de crear el Unbounded Task Flow se debe exportar como un ADF Menu, que es un simple archivo XML con una cierta estructura. Con la creación del ADF Menu ya se podría crear un componente que siguiera la estructura definida directamente en una página JSF, pero duplicaría el código en todas las páginas en las que queramos insertar el menú (yendo en contra de la reutilización de código). Lo más recomendable en estos casos es crear una plantilla que puede ser utilizada cuando se creen las páginas más adelante, con lo que no solo el código no se repetiría sino que además en el caso de añadir algún cambio en el menú solo habría de realizarse en la plantilla y el cambio se heredaría en todas las páginas que la utilicen.

La página Configuración según los requisitos permite la configuración de los elementos que se van a utilizar en la aplicación (Clientes, Proyectos, Tareas, Empleados, Calendarios, Organizaciones, Centros de Trabajo, Grupos de Trabajo y Horarios). Aunque esto se podría realizar agregando hijos al nodo configuración del menú ADF creado antes y añadiéndolo a la plantilla, se ha optado por realizar una región dinámica a través de Bounded Task Flows con el fin de mejorar el conocimiento

de éstos. Para desarrollar una región dinámica hay que seguir los siguientes pasos, en primer lugar crear un Bounded Task Flow por cada "flujo" que intervenga en la región. En este caso el "flujo" es simplemente mostrar un fragmento de página por lo que dentro de cada Bounded Task Flow se encontrará el fragmento. El siguiente paso es crear la región dinámica, la opción más sencilla es arrastrar un de los task flows a la página donde se quiera agregar la región dinámica (en este caso Configuración) ya que JDeveloper genera los bindings y el Backing Bean que controla la región. Por último se deben agregar los demás task flows como enlaces de región dinámica, por lo que es recomendable utilizar algún tipo de componente donde la interfaz se separe en dos (tipo af:panelSplitter), en el que una parte contenga los enlaces y la otra parte la región dinámica en sí.

4.2.4. Seguridad

La seguridad en una aplicación web es uno de los aspectos más importantes, recoge la autenticación y la autorización de los usuarios. En los requisitos de la aplicación AriNet 2, todos los usuarios tienen que estar autenticados para poder acceder a cualquier página y posteriormente debe limitarse la autorización de la página Configuración. La autenticación debe realizarse contra un Active Directory (LDAP de Microsoft), del que se recogen los usuarios y los grupos.

En primer lugar se deberá configurar el servidor Weblogic para que conecte con el Active Directory donde se encuentran los usuarios y grupos que intervendrán en la autenticación y autorización. Esto se realizará directamente en la aplicación de administración del servidor de aplicaciones Weblogic, donde se debe crear un nuevo proveedor de autenticación con los datos de conexión del Active Directory.

Una vez el servidor Weblogic está correctamente configurado, hay que configurar la seguridad de ADF dentro de la aplicación. En este caso para cumplir los requisitos de seguridad, ADF Security ha sido configurada para autenticación y autorización. El tipo de autenticación seleccionado es basado en formulario, aunque podría haberse elegido los otros tipos con el mismo resultado. Por último la autorización, ésta puede ser realizada de distintas formas pero la seleccionada ha sido "*Grant to All Objects*", ya que en los requisitos se ha decidido que los usuarios autenticados pueden acceder a todas las páginas excepto a Configuración, que está limitada al grupo "Administrador" recuperado del Active Directory por medio del Weblogic.

Con la seguridad configurada hay que dar permisos únicamente a los usuarios del grupo "Administrador" de Weblogic para poder acceder a la página Configuración, para ello hay que crear un "*Enterprise Rol*" que se mapee con el grupo del Weblogic y un "*Application Rol*" que se mapee con el "*Enterprise Rol*" anterior, y concederle los

permisos de la página Configuración.

5. Conclusiones

A continuación, se describirán las conclusiones a las que se ha llegado una vez terminado el proyecto de fin de grado, junto con las mejoras que se incluirán en futuras versiones del complemento y el presupuesto.

5.1. Conclusiones

Tras la finalización de las tareas del presente trabajo de fin de grado, es momento de valorar resultados y sacar conclusiones. Tal y como se recogía en el apartado inicial, el objetivo fundamental era el análisis del framework de desarrollo de aplicaciones Java EE Oracle ADF. A lo largo de este documento se ha descrito la estructura de ADF Framework haciendo hincapié en sus ventajas frente a otros frameworks.

Además, también se han logrado otros objetivos secundarios:

Comparar el desarrollo de aplicaciones en el framework ADF respecto a su antigua versión (OAF framework).

La primera diferencia que hay que indicar entre OAF y ADF es el objetivo de cada una de ellos. ADF está pensado para crear aplicaciones empresariales completas, sin embargo el objetivo de OAF era pasar parte de la lógica de E-Bussiness Suite a la web.

Además estructuralmente son muy diferentes, sólo coinciden en los ADF Bussiness Components, que han sido llevados de OAF a ADF.

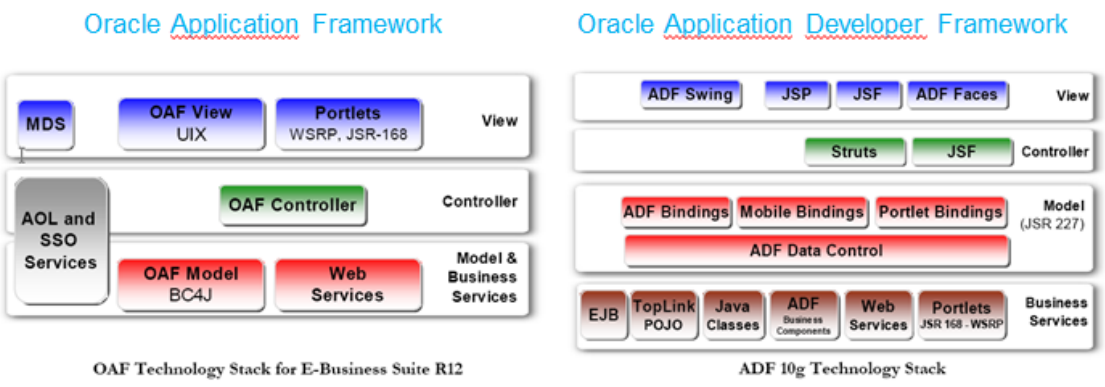

Ilustración 53. OAF vs ADF

Descubrir el esfuerzo que se necesita para aprender a desarrollar aplicaciones en este framework.
El aprendizaje del framework ADF para el desarrollo de aplicaciones Fusion no es una tarea sencilla. Al comenzar es necesario asimilar una gran cantidad de conceptos en cuanto a las diferentes capas para tener una visión global de cómo interactúan las capas entre ellas, pero una vez se aprenden, el ritmo de aprendizaje aumenta de manera de exponencial.

Desarrollar una prueba de concepto, con el fin de afianzar los conocimientos resultantes del análisis de ADF y la aplicación práctica de los mismos.
Se ha desarrollado una aplicación empresarial diseñando capa a capa teniendo en cuenta las capacidades que ofrece cada una. Además durante la implementación se han descubierto nuevas capacidades que a priori no se conocen y sobretodo la facilidad de desarrollo que ofrece JDeveloper.

Realizar una toma de contacto con Weblogic, que es el servidor de aplicaciones por defecto de ADF y al que se están migrando las aplicaciones Oracle.
Durante la implementación de la prueba de concepto se han necesitado realizar ciertas configuraciones sobre el servidor de aplicaciones Weblogic:

- Instalación de un servidor Weblogic independiente en una máquina Linux y su posterior creación del dominio y las máquinas que utiliza.
- Instalación de las librerías de ADF que dependen de la versión de JDeveloper y de ADF Runtime utilizadas.
- Configuración de un clúster con distintas máquinas para mejorar la disponibilidad de la aplicación.
- Configuración de un origen de datos.
- Creación de un nuevo proveedor de autenticación.

Por tanto, se puede afirmar que el proyecto ha sido un éxito, tras cumplir el objetivo principal y los secundarios.

5.2. Futuros trabajos

En todo software informático complejo se van a encontrar bugs y especialmente en este tipo de sistemas. Aun así independiente de los bugs existentes, todo es mejorable, y muestra de ello es el número de versionas que tiene ADF, en las que mejora o integra nuevas características para satisfacer las necesidades de los

desarrolladores.

A continuación se recogen diferentes aspectos que puedan ser mejorados o ampliados en futuras versiones:

Escasez de documentación exhaustiva

La naturaleza declarativa de ADF causa que las guías para los desarrolladores muestran "el cómo" y no "el por qué", por lo que los desarrolladores de Oracle ADF son obligados a mirar el código fuente que además generalmente está ofuscado.

Proveer un kit de desarrollo para desarrolladores ASP.NET

Actualmente ADF es un framework de desarrollo J2EE, pero sería una alternativa interesante extrapolarlo al desarrollo de aplicaciones ASP.NET, que cuentan también con una gran cuota de mercado.

Reducir el consumo de recursos del JDeveloper durante el desarrollo

El JDeveloper consume una gran cantidad de recursos del sistema y más aún cuando se utiliza el servidor Weblogic integrado. Esto obliga a tener un ordenador muy potente para los desarrolladores ADF.

Ampliar la comunidad de desarrolladores ADF

Durante el desarrollo de software, y más durante el aprendizaje de un lenguaje o framework nuevos, es muy importante buscar información. La comunidad de desarrolladores de ADF es muy reducida, y esto supone que la información disponible también lo sea.

Soportar diferentes alternativas a Android e iOS en ADF Mobile

En la actualidad ADF Mobile sólo soporta Android e iOS como sistemas operativos móviles a los que producir las aplicaciones. Aunque sean los que tienen mayor cuota de mercado, la importancia de otros (como Windows Phone) está aumentando y por ello deben ser tenidos en cuenta para un futuro.

5.3. Conclusión Personal

Como conclusión quiero manifestar mi opinión e impresiones acerca del trabajo realizado. Desde mi punto de vista, Oracle ADF es una opción más que recomendable tanto para aquellas empresas que quieran realizar aplicaciones empresariales con J2EE como para empresas ligadas con los productos de Oracle que quieran implantar

Fusion dentro de su corporación y desarrollar aplicaciones para ello.

Además JDeveloper agiliza increíblemente el desarrollo de una gran cantidad de funcionalidades en cada una de las capas y para las funcionalidades no preconstruidas siempre se puede realizar el desarrollo mediante código Java, que es el lenguaje orientado a objetos más utilizado por millones de dispositivos y como tal cuenta con un gran número de desarrolladores. A su vez, el uso de ADF BC en la capa de servicios de negocio facilita todo lo relacionado con los orígenes de datos (conexión, vista, edición e inserción de datos, etc…)

Nada en esta vida es perfecto y como mayor problema he de decir que el aprendizaje (sobre todo al comienzo, en los primeros pasos con el framework) es una tarea muy dura cuando se realiza de manera autodidacta. Es cierto que Oracle tiene una gran cantidad de información, pero ésta es muy densa, con un lenguaje complicado y de una manera muy teórica. Además la comunidad de desarrolladores de ADF Framework es muy reducida por lo que foros y demás sitios donde buscar distintos casos de uso y problemas ya encontrados carecen utilidad alguna. Por lo que hasta que un desarrollador empieza a dominar el marco de trabajo puede pasar bastante tiempo.

En cuanto a mi motivación personal, este Trabajo de Fin de Grado me ha sido de gran ayuda para adquirir un cierto dominio sobre la estructura de las aplicaciones J2EE, que hasta el momento de iniciación del trabajo, era muy reducido. Además supone una nueva habilidad que poder desarrollar en mi carrera laboral y que me abre las puertas a un nuevo mundo en expansión, que es el desarrollo de aplicaciones empresariales J2EE.

Por último, añadir que este trabajo recoge únicamente las características principales de este framework para facilitar el aprendizaje del mismo, pero ADF Framework ofrece un sinfín de características que aquí no están explicadas y que pueden ser realizadas.

6. Planificación y Presupuesto

6.1. Planificación

Para la correcta ejecución del Trabajo Fin de Grado y la estimación de plazos, recursos y presupuesto, se hace un estudio inicialmente y se estructuran las tareas. Para una gestión eficiente, las tareas más heterogéneas han sido divididas en subtareas. A continuación se muestran las tareas planificadas inicialmente en una tabla y en un diagrama de Gantt.

Tarea	Subtareas	Descripción
Análisis del framework	• Búsqueda de Información • Estudio y asimilación de la información	Aprendizaje teórico del framework y análisis del mismo
Iniciación con JDeveloper		Toma de contacto con ADF a través de ejemplos disponibles en la web de Oracle
Prueba de Concepto	• Diseño con ADF • Planificación • Desarrollo	Desarrollo de la prueba de concepto con ADF
Redacción de la memoria del TFG		Documentación del trabajo realizado

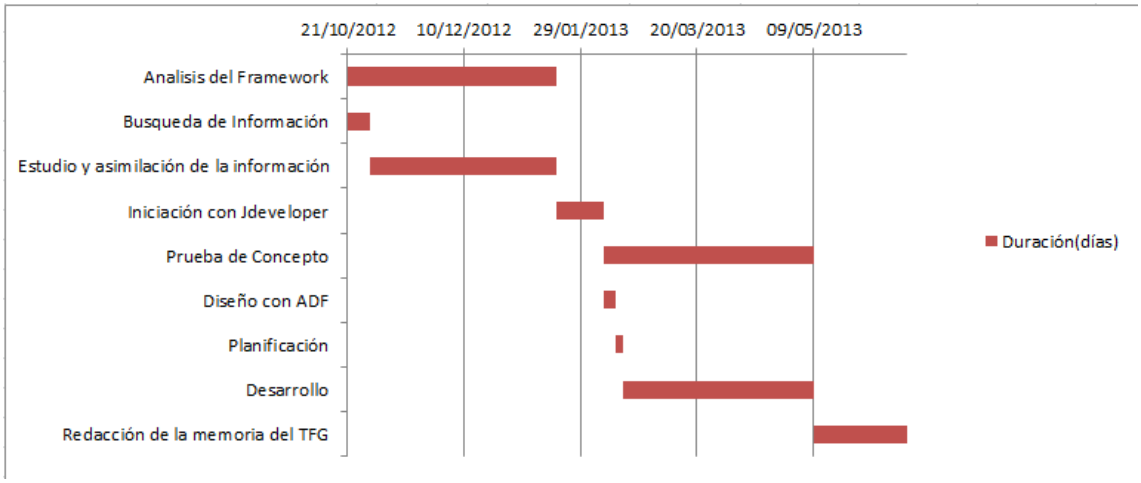

Ilustración 54. Diagrama de Gantt

6.2. Presupuesto

A continuación se expone el presupuesto calculado para la realización de este Trabajo Fin de Grado. Para el cálculo se ha como referencia la planificación del punto 6.1 con un total de 242 días a una media de 20-25 horas semanales, los costes de personal pueden observarse en la siguiente tabla.

Puesto	Horas Totales	Coste/hora	Coste Total
Desarrollador Junior	800	20	16.000 €
Recursos Humanos			16.000€

Dado que el proyecto ha sido desarrollado con fines académicos y de investigación no se han aplicado porcentajes extra por riesgos y beneficios en los costes del personal. Para los recursos hardware y software se ha calculado la amortización en función de los meses en los que han sido utilizados.

Elemento	Coste	Tiempo de Vida	Coste Mensual	Meses Utilizado	Coste Amortizado
Hardware					
HP ProBook 6065b	800€	36	22€	9	200€
HP Pavilion p6-2304es	500€	36	14€	9	125€
HP Pavilion p6-2304es	500€	36	14€	9	125€
Coste Hardware					450€
Software					
Licencia Windows 7	164€	50	3€	9	27€
Licencia Microsoft Office 2010	199€	50	4€	9	36€
Licencia Weblogic Server Standard Edition y ADF Runtime	158€	50	3€	9	27€
Coste Software					90€
Coste Total					540€

Elemento	Coste Total
Recursos Humanos	16.000€
Hardware y Software	540€
Coste Total	16.540€

La suma total del presupuesto asciende a una cuantía final de DIECISEIS MIL QUINIENTOS CUARENTA EUROS.

7. Anexo I: Requisitos de la prueba de concepto

A continuación se enumeran y detallan los diferentes requisitos que la prueba de concepto deberá cumplir.

A modo informativo, las tablas de requisitos tienen la siguiente estructura:

- **Identificador:** campo unívoco para identificar cada requisito.
- **Tipo:** Se distinguirá entre funcional o no funcional.
  - Funcional: especifican el qué tiene que hacer el software.
  - No Funcional: imponen restricciones en el diseño.
- **Necesidad:** Se distinguirá entre esencial, deseable y opcional.
  - Esencial: de necesidad obligatoria en el trabajo a realizar.
  - Deseable: de carácter negociable
  - Opcional: la no inclusión no supone penalización.
- **Descripción:** breve explicación del contenido del requisito.

Identificador: RF-01	
Tipo	Funcional
Necesidad	Esencial
Descripción	La aplicación web tiene que tener 4 páginas: Inicio, Mis Imputaciones, Informes y Configuración

Identificador: RF-02	
Tipo	Funcional
Necesidad	Esencial
Descripción	Debe haber un menú para la navegación entre páginas

Identificador: RNF-03	
Tipo	No Funcional
Necesidad	Esencial
Descripción	Las páginas sólo deben ser visibles a usuarios autenticados, a la página Informes sólo los usuarios que estén en el grupo MANAGER, y a la página Configuración sólo pueden los usuarios que estén en el grupo ADMINISTRADOR

Identificador: RF-04	
Tipo	Funcional
Necesidad	Deseable
Descripción	Se debe mostrar el nombre del usuario autenticado

Identificador: RNF-05	
Tipo	No Funcional
Necesidad	Esencial
Descripción	La autenticación de usuarios debe realizarse contra un LDAP con los datos de los usuarios.

Identificador: RF-06	
Tipo	Funcional
Necesidad	Esencial
Descripción	En la página de Inicio se deben mostrar los días pendientes de imputación del usuario autenticado.

Identificador: RF-07	
Tipo	Funcional
Necesidad	Esencial
Descripción	En la página Mis Dedicaciones se debe poder imputar las horas que realiza en su jornada laboral y asignárselas a un cliente, a un proyecto y a una tarea

Identificador: RF-08	
Tipo	Funcional
Necesidad	Esencial
Descripción	Los Clientes, Proyectos y Tareas a los que se pueden imputar horas deberán estar filtrados para la organización y grupo de trabajo al que pertenece el usuario autenticado

Identificador: RF-09	
Tipo	Funcional
Necesidad	Esencial
Descripción	En la página Informes de deben poder sacar informes de las imputaciones realizadas por los empleados y deben poder exportarse a Excel

Identificador: RF-10	
Tipo	Funcional
Necesidad	Esencial
Descripción	En la página Configuración debe haber un submenú con 9 páginas organizado en 3 pestañas para realizar la configuración necesaria sobre la aplicación: • Imputaciones: Clientes, Proyectos y Tareas • Empleados: Empleados y Calendarios • General: Organizaciones, Centros de Trabajo, Grupos de Trabajo y Horarios

Identificador: RF-11	
Tipo	Funcional
Necesidad	Esencial
Descripción	En cada una de las páginas de dentro de Configuración debe haber un buscador con distintos parámetros de entrada.

Identificador: RF-12	
Tipo	Funcional
Necesidad	Esencial
Descripción	En cada una de las páginas de dentro de Configuración debe haber una tabla donde se muestren los resultados que concuerden con la búsqueda.

Identificador: RF-13	
Tipo	Funcional
Necesidad	Esencial
Descripción	En cada una de las páginas de dentro de Configuración debe haber una región donde se muestre los campos de un único registro.

Identificador: RF-14	
Tipo	Funcional
Necesidad	Esencial
Descripción	En cada una de las páginas de dentro de Configuración se debe poder insertar nuevos registros.

Identificador: RF-15	
Tipo	Funcional
Necesidad	Esencial
Descripción	En la región donde se muestre la información de un único registro de la página de Configuración debe haber botones para la navegación a distintos registros (siguiente, anterior, primero y último).

Identificador: RF-16	
Tipo	Funcional
Necesidad	Esencial
Descripción	En la tabla que muestra los resultados de la búsqueda de la página de Configuración se debe permitir seleccionar un registro para mostrar su información en la región de visión de la información de los registros.

8. Anexo II: Manual de Usuario

8.1. Autenticación

Lo primero de todo, gracias a haber configurado la seguridad de ADF es autenticarse, es igual la página a la que se intente acceder que se redirigirá al usuario a la página de autenticación.

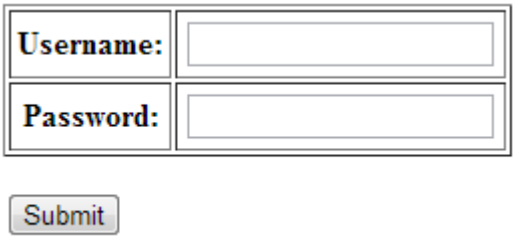

Ilustración 55. Página de autenticación

En este caso es la página que genera por defecto la seguridad de ADF pero podría haber sido cambiada por cualquier otra siempre que el action del formulario sea "*j_security_check*".

8.2. Inicio

La página de Inicio es a la página a la que se redirigirá al usuario, una vez la autenticación haya sido correcta. En ella se encuentra un menú de navegación a las distintas páginas (Inicio, Mis Dedicaciones, Informes y Configuración) y además en el caso de que haya dedicaciones pendientes de validar o de imputar muestra un aviso.

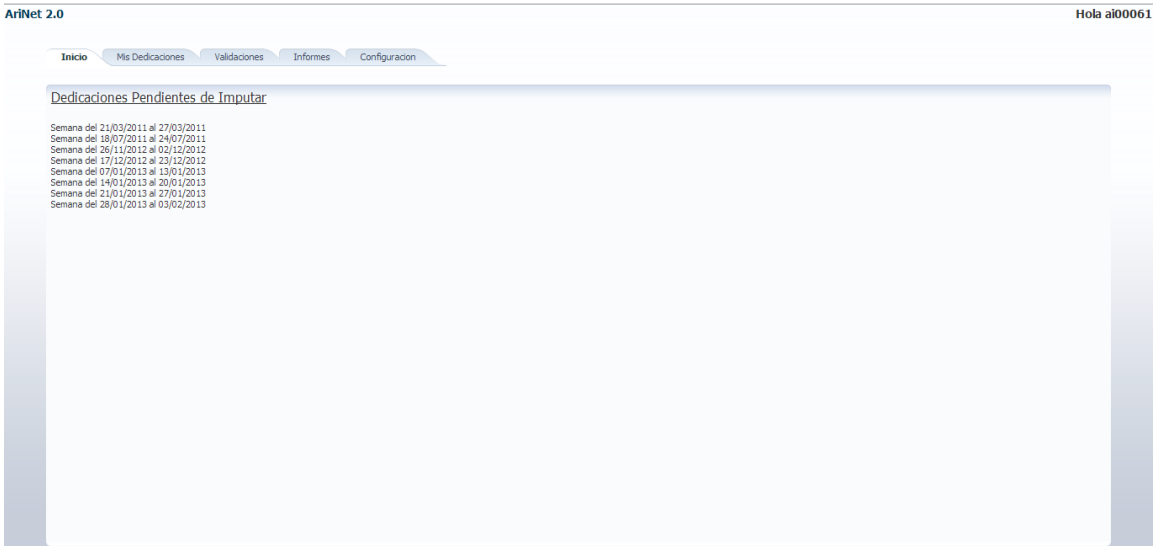

Ilustración 56. AriNet 2 - Página de Inicio

8.3. Mis Dedicaciones

En la página Mis Dedicaciones los empleados pueden imputar las horas que han realizado durante su jornada laboral.

En la imagen se ve un acordeón en la parte izquierda y un calendario en la derecha. El acordeón tiene todos los clientes, proyectos y tareas a los que el usuario autenticado puede imputar horas.

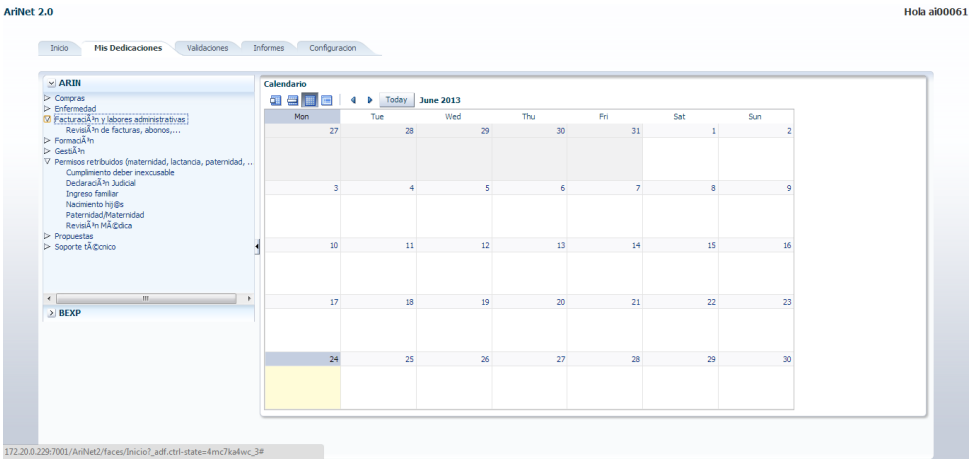

Ilustración 57. AriNet 2 - Mis Dedicaciones

Para realizar una imputación el usuario puede hacer click sobre un día concreto, con lo que se abrirá un dialogo donde se seleccionará el cliente, el proyecto y la tarea con el día y la hora establecidos (la hora se establecerá a las 9.00 de la mañana si no hay tarea previa en ese día y si hay un tarea imputada ese día la hora establecida será la hora de finalización de la última tarea del día donde se ha hecho click).

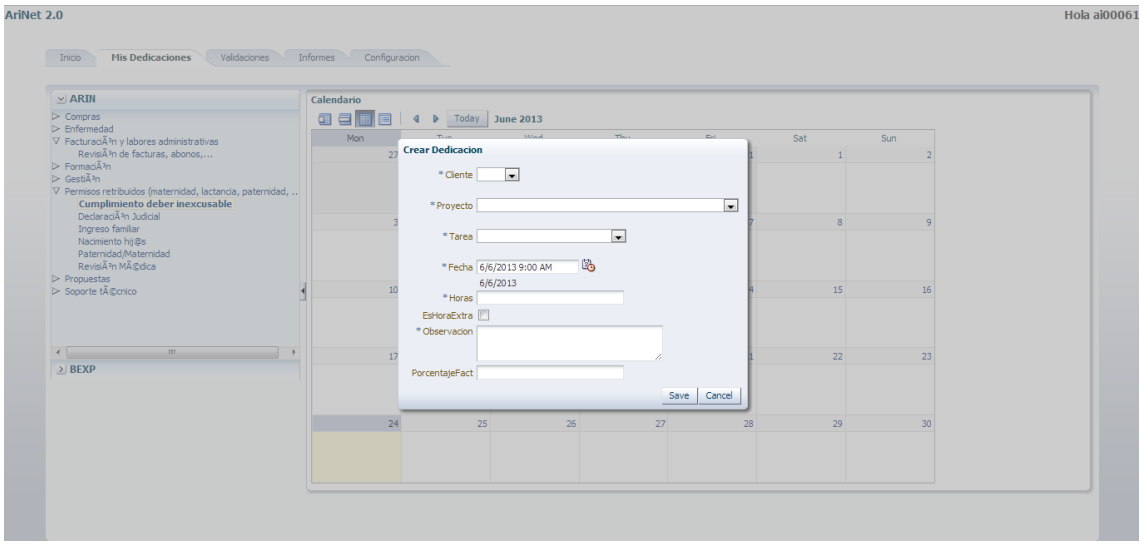

Ilustración 58. AriNet 2 - Imputar una tarea

El usuario también tiene la opción de utilizar el menú de clientes de la izquierda para buscar la tarea a la que quiera imputar las horas y la arrastre a un día concreto del calendario, con lo que saldrá un dialogo con el cliente, proyecto y tarea ya establecidos.

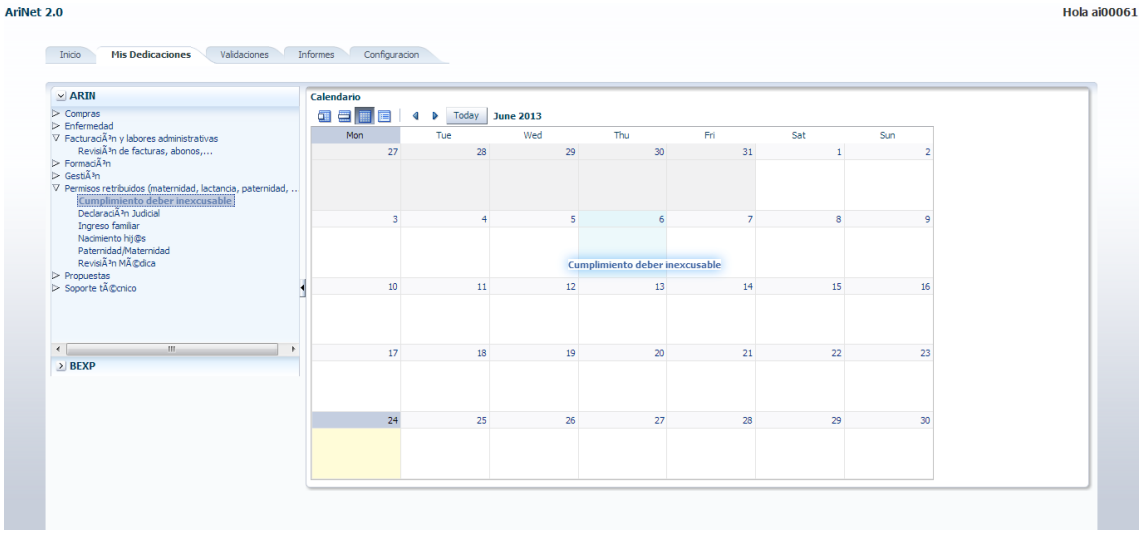

Ilustración 59. AriNet 2 - Drag and drop de una tarea

Una vez se hayan imputado una tarea, puede ser editada o eliminada y para ello sólo hay que hacer click con el botón secundario sobre la imputación realizada y elegir la opción que se quiera realizar del menú contextual.

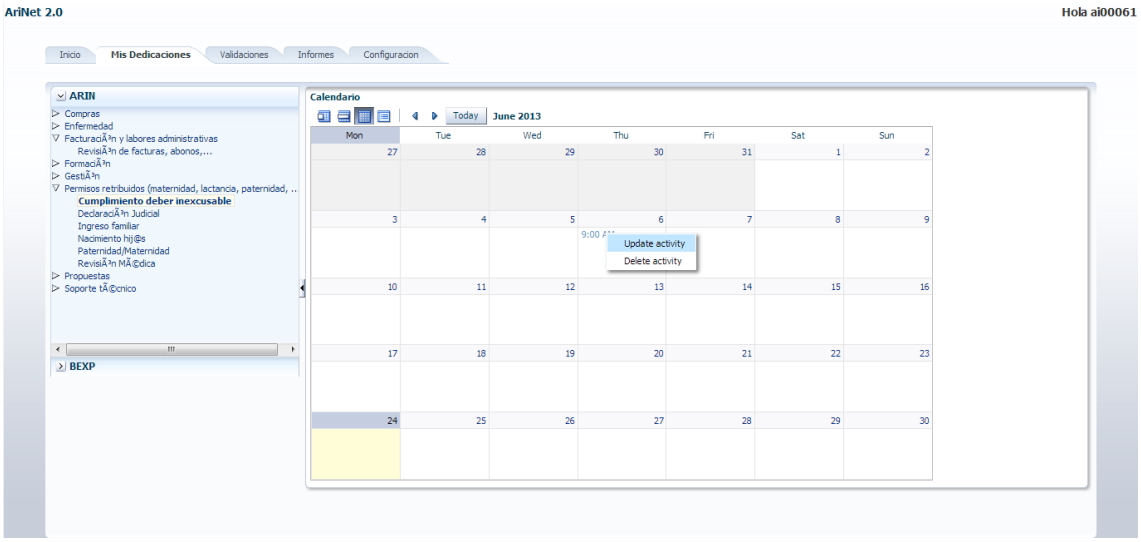

Ilustración 60. AriNet 2 - Menu contextual

8.4. Informes

En la página Informes pueden realizar consultas de las imputaciones realizadas por todos los empleados, buscando en función de distintos parámetros (usuario, cliente, proyecto, tarea, día, etc…).

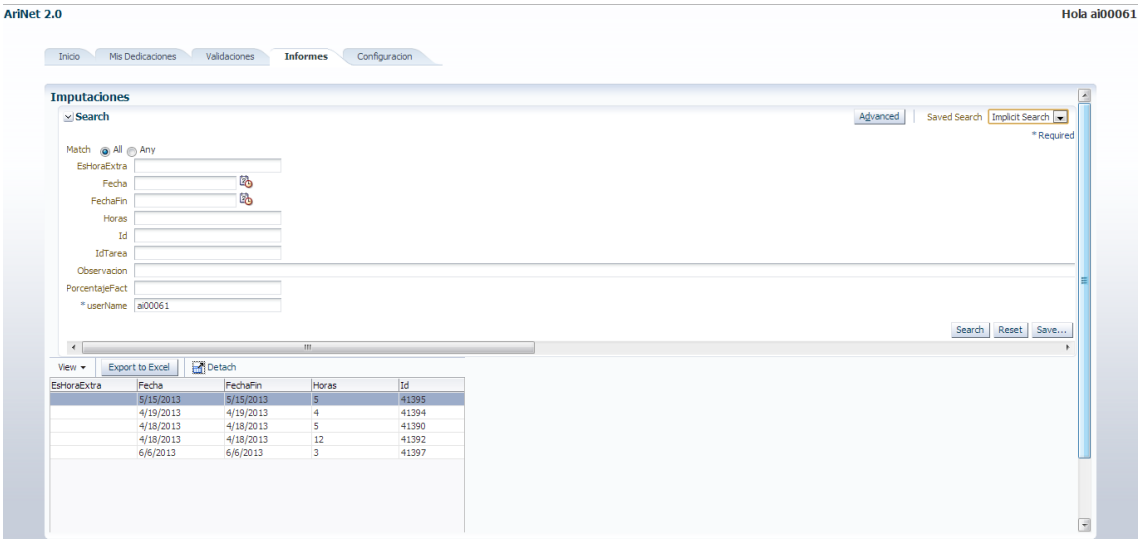

Ilustración 61 - AriNet 2 – Informes

Gracias a las funcionalidades que los componentes de ADF Faces nos proporcionan se pueden realizar todo tipo de operaciones sobre la tabla, como filtrar, ordenar e incluso exportar a Excel los datos obtenidos.

8.5. Configuración

En la página Configuración se pueden realizar todos los ajustes necesarios para la puesta en marcha de la aplicación. En ésta se tiene una región dinámica y un menú a la izquierda donde están todos los links que actualizan la región dinámica al contenido que el usuario decida.

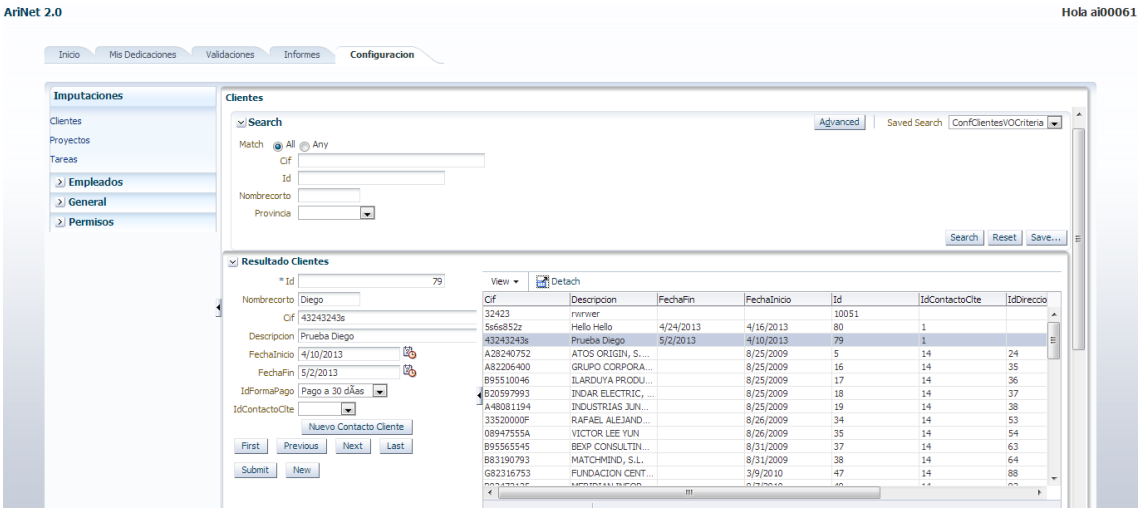

Ilustración 62. AriNet 2 – Configuración

Todas las páginas son similares en cuanto a interfaz y funcionalidad, por lo que se va a explicar únicamente lo común a todas debido al gran número de contenido de esta página.

En primer lugar hay un buscador, generalmente tendrá los campos del View Object en el que se quiera buscar, pero en algunos casos se han agregados otros campos calculados o incluso se han creado listas de valores. Este buscador está realizado con el componente af:query que permite buscar por ningún, algún o todos los campos utilizados y guardar las búsquedas realizadas.

Además del buscador hay otra región separada en dos para mostrar los resultados de la búsqueda realizada. En la parte de la derecha se encuentra con una tabla con todos los registros resultantes de la búsqueda realizada, en ella se mostrarán

los campos que sean necesarios y el usuario puede editar la tabla para filtrar, ordenar o quitar campos de la tabla. En la otra parte se muestran todos los campos del registro que está puesto como actual en el iterador del View Object, para explicar mejor el porqué de esto, se va a mostrar un ejemplo.

Supongamos que se quiere ver o editar los datos de un determinador registro que no se conoce, simplemente algún campo de él. Para ello iremos al buscador e introduciremos los datos conocidos (si los hay) y se hará click en el botón de buscar, con lo que automáticamente se rellenará la tabla de abajo con los registros que cumplan los parámetros de búsqueda. Con esto tendremos todos los registros en la tabla y si hacemos click en alguno de ellos se cambiará el registro actual del iterador del View Object a ese registro y por tanto se rellenará la parte de la izquierda con los datos del registro actual, con lo que pueden ser observados de una forma más sencilla e incluso editados por los deseados. Además se permite crear nuevos registros y guardar todos los cambios que se realicen.

9. Bibliografía

[1] Oracle ADF Real World Developer's Guide: Mastering essential tips and tricks for building next generation enterprise application with Oracle ADF. Jobinesh Purushothaman, Octubre 2012

[2] Oracle Fusion Developer Guide: Building Rich Internet Applications with Oracle ADF Bussiness Components and Oracle ADF Faces. Frank Nimphius y Linn Munsinger, 2010

[3] Oracle® Fusion Middleware: Fusion Developer's Guide for Oracle Application Development Framework 11g Release 2, Mayo 2011

[4] Quick Start Guide to Oracle Fusion Development: Oracle Jdeveloper and Oracle ADF. Grant Ronald, 2011

[5] Oracle® Fusion Middleware: Web User Interface Developer's Guide for Oracle Application Development Framework 11g Release 1, Octubre 2009

[6] Oracle ADF 11gR2 Development Beginner's Guide. Vinod Krishnan, Abril 2013

[8] Oracle JDeveloper 11g Handbook: A Guide to Oracle Fusion Web Development. Duncan Mills; Peter Koletzke; Avrom Roy-Faderman, Octubre 2009

[7] Oracle ADF Enterprise Application Development—Made Simple. Sten E. Vesterli, Junio 2011

[8] Overview of Enterprise Applications - Your First Cup: An Introduction to the Java EE Platform. Disponible:
http://docs.oracle.com/javaee/6/firstcup/doc/gcrky.html

[9] N-Tier Application Architecture. Disponible:
http://www.webopedia.com/quick_ref/app.arch.asp

[10] Introducing Enterprise Java Application Architecture and Design. Disponible: http://www.developer.com/design/article.php/3808106/Introducing-Enterprise-Java-Application-Architecture-and-Design.htm

[11] Struts 1 – Introduction. Disponible:
http://struts.apache.org/development/1.x/userGuide/introduction.html

[12] SpringSource – Documentation. Disponible:
http://www.springsource.org/documentation

[13] Spring Framework Architecture. Disponible:
http://www.tutorialspoint.com/spring/spring_architecture.htm

[14] Seam Framework - JBoss Seam. Disponible:
http://www.seamframework.org/

[15] Developer's Guide - Google Web Toolkit. Disponible:
https://developers.google.com/web-toolkit/doc/latest/DevGuide

[16] Oracle Application Development Framework Overview. Disponible: http://www.oracle.com/technetwork/developer-tools/adf/adf-11-overview-1-129504.pdf

[17] Configuring ADF Security. Disponible: http://one-size-doesnt-fit-all.blogspot.com.es/2010/07/adf-security-revisited-again-again.html

[18] ADF Training | How to Create Navigation Panes using a Menu Model in Oracle ADF. Disponible: http://www.youtube.com/watch?v=cInX_FMZJaQ